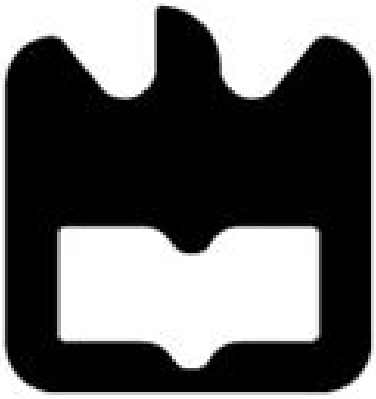
Universidade de Aveiro
2009

Departamento de Electrónica, Telecomunicações e Informática

Fernando de Almeida e Costa

Desenvolvimento de Centralina Automóvel com Arquitectura Distribuída



Universidade de Aveiro
2009

Departamento de Electrónica, Telecomunicações e Informática

Fernando de Almeida e Costa

Desenvolvimento de Centralina Automóvel com Arquitectura Distribuída

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Manuel Bernardo Salvador Cunha, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Eu amo o Longe e a Miragem,
Amo os abismos, as torrentes, os desertos...
(…)
Não sei por onde vou,
Não sei para onde vou
- Sei que não vou por aí!

José Régio, Cântico Negro *in "Poemas de Deus e do Diabo"*

o júri

presidente

Professora Doutora Ana Maria Perfeito Tomé

Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais

Professor Doutor Manuel Bernardo Salvador Cunha

Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

Professor Doutor José Luís Costa Pinto de Azevedo

Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Co-Orientador)

Professor Doutor António Paulo Gomes Mendes Moreira

Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

agradecimentos

Os meus sinceros agradecimentos ao Professor Doutor Bernardo Cunha pela dedicação e orientação prestadas ao longo desta dissertação, e sobretudo pela motivação e os vastos conhecimentos oferecidos. Agradeço-lhe igualmente a oportunidade de ter contribuído com os meus esforços e conhecimentos para o projecto ICARO.
Ao Engenheiro David Ribeiro pelo seu apoio, companheirismo e opiniões oportunamente partilhadas; ao IEETA e às pessoas que diariamente me acompanharam durante o projecto; e à Universidade de Aveiro pelo curso que finalizo: o meu encarecido apreço.

Não esquecerei a amizade, a alegria, a compreensão e a estima de todos os meus amigos e colegas que fizeram parte desta etapa e contribuíram para a pessoa que hoje sou.
Pessoalmente, a todos vós, um grande Obrigado e um até sempre…

Por fim e acima de tudo, agradeço profundamente ao meu pai Fernando, à minha mãe Maria Cândida e às minhas irmãs Andresa e Linda por toda a dedicação que me presentearam e aquilo que representam na minha vida como família. Sem vós não teria chegado onde cheguei.

palavras-chave

Sistemas Distribuídos, Controlo Electrónico Automóvel, Protocolos de Comunicação

resumo

O desenvolvimento de uma centralina automóvel com arquitectura distribuída enquadra-se no projecto ICARO (http://icaro.ua.pt). Este projecto caracteriza-se por ser uma iniciativa de carácter pluridisciplinar da Universidade de Aveiro, envolvendo áreas como mecânica, design, electrónica, sistemas de informação e informática. Tem como propósitos a participação anual de uma equipa representante da Universidade de Aveiro na prova Europeia da *Shell Eco-marathon* (www.shell.com/eco-marathon), e a potenciação do desenvolvimento e inovação nas áreas associadas. Sendo o objectivo principal da competição a poupança de combustível, e estando em fase de conclusão a construção de um novo veículo, pretende-se agora desenvolver uma nova centralina para o motor de combustão interna baseada num paradigma distribuído.

O tema desta dissertação enquadra-se na implementação da solução distribuída para a centralina automóvel, desenvolvendo e testando os módulos constituintes. A solução prevê a implementação de unidades independentes para a execução das várias tarefas de controlo do motor. Desta forma, espera-se alcançar uma optimização dos tempos de resposta nos elementos principais de controlo do motor, proporcionando liberdade de configuração e ajuste de forma independente.
Foi ainda desenvolvida uma Interface de Monitorização e Diagnóstico para acompanhamento em tempo real das tarefas de controlo, assim como a reconfiguração de parâmetros do motor. Além disso, a utilização de valores tabelados para o controlo da injecção e ignição requer a existência de uma interface entre a ECU e a unidade computacional que possui essa informação.

keywords

Distributed Systems, Automotive Electronic Control, Communication Protocols

abstract

The development of an Electronic Control Unit (ECU) based on a distributed system relates with the ICARO project (http://icaro.ua.pt). This project is a multidisciplinary initiative from Universidade de Aveiro, involving several areas as mechanics, design, electronics, information systems and computer science. It has the purpose of the annual team participation at the European Shell Eco-marathon competition (www.shell.com/eco-marathon), and simultaneously encouraging the development and innovation. Considering that the competition's main goal is to reduce fuel consumption, and the existence of a new almost finished race vehicle, the main objective of this work is to develop an innovative distributed ECU system to control the vehicle's internal combustion engine operation.

The theme of this thesis encompasses the development of a distributed ECU system through the implementation and testing of its different modules. It is intended to improve the tasks' execution times, taking benefit from the control modules' independent operation.
The development of a Monitoring and Debug Interface is included with the objective of real-time motor control tasks attendance, providing extra features such as communication between the ECU and a computer for data transmission.

Conteúdo

Conteúdo ..... **i**

Índice de Figuras ..... **iii**

Lista de Acrónimos ..... **v**

1. Introdução e Estado da Arte **1**

1.1. Estrutura do Documento **1**

1.2. Motivação e Objectivos **2**

1.3. Sistemas de Tempo-Real **3**

1.4. Sistemas Distribuídos **4**

1.4.1. Caracterização **4**

1.4.2. Solução Distribuída vs. Solução Centralizada **5**

1.5. Protocolos de Comunicação no Contexto Automóvel **8**

1.5.1. LIN – Local Interconnect Network **14**

1.5.1.1. Características Técnicas **14**

1.6. Controlo Electrónico Automóvel **17**

1.6.1. Controlo Electrónico de Motores de Combustão Interna **17**

1.6.1.1. Emissões de Escape, Consumo de Combustível e Força Produzida **18**

1.7. Motor do Ícaro – Controlo Electrónico **21**

1.7.1. Princípio de Funcionamento **21**

1.7.2. Sensores e Actuadores Utilizados **23**

2. Desenvolvimento da Solução Distribuída **25**

2.1. Migração do Sistema Centralizado para o Sistema Distribuído **25**

2.1.1. Módulo de Controlo Central (*Master*) **28**

2.1.2. Módulo de Controlo da Pressão do Combustível **29**

2.1.3. Módulo de Controlo da Injecção **31**

2.1.4. Módulo de Controlo da Ignição **33**

2.1.5. Canal de Sincronização Dedicado 35
2.1.5.1. Características Técnicas 36
2.1.5.2. Solução Implementada 38

3. Implementação Ícaro LIN 39
3.1. Adaptações Específicas do Protocolo ao Projecto 39
3.2. Solução Implementada 41
3.2.1. LIN *Master Task* 41
3.2.2. LIN *Slave Task* 42
***3.2.2.1.* Implementação *Master* 43**
***3.2.2.2.* Implementação *Slave* 46**

4. Interface de Monitorização e Diagnóstico 49
4.1. Comunicação USB 49
4.1.1. Módulo Conversor USB-Paralelo 50
4.1.2. Características Técnicas do Protocolo Implementado 52
4.1.3. Solução Implementada 53
4.2. Interface Gráfica 56

5. Resultados e Discussão 59

6. Conclusões e Desenvolvimentos Futuros 65

Referências Bibliográficas 69

Índice de Figuras

Figura 1 – Evolução das redes de comunicação no contexto automóvel baseado nos modelos W210 (1995) e W211 (2002) da série Classe E da Mercedes-Benz. 9

Figura 2 – Representação das diversas redes actualmente existentes em veículos automóveis. 12

Figura 3 – Representação da aplicabilidade do protocolo LIN no ambiente automóvel.....13

Figura 4 – Esquema representativo da configuração de uma rede LIN. 15

Figura 5 – Exemplo da LIN *frame*, identificado cada campo descrito anteriormente.....16

Figura 6 – Efeito de lambda nas emissões dos gases de escape, torque produzido e consumo de combustível. 19

Figura 7 – Efeito do *timing* de ignição nas emissões dos gases de escape, torque e consumo de combustível. 20

Figura 8 – Esquema ilustrativo das fases de um ciclo completo num motor de explosão a quatro tempos. 22

Figura 9 – Representação esquemática do motor, sensores, actuadores e ECUs previstos para o controlo electrónico do mesmo. 27

Figura 10 – Aspecto real do Módulo de Controlo Central com as unidades descritas assinaladas. 28

Figura 11 - Aspecto real do Módulo de Controlo Central com as unidades descritas assinaladas. 30

Figura 12 - Aspecto real anterior do Módulo de Controlo da Injecção com as unidades descritas assinaladas. 31

Figura 13 - Aspecto real posterior do Módulo de Controlo da Injecção com as unidades descritas assinaladas. 32

Figura 14 - Aspecto real do Módulo de Controlo da Ignição com as unidades descritas assinaladas. 34

Figura 15 – Representação do sinal gerado pelo Sensor Angular de Indução e a numeração correspondente de cada ponto de referência 36

Figura 16 – Representação do sinal de sincronização final, a numeração dos pontos de referência e a indicação correspondente a cada fase de um ciclo. 37

Figura 17 – Representação da LIN *frame* com as alterações descritas. 40

Figura 18 – Fluxograma da função *ProcessLinMater()* executada na LIN *Master Task*. 45

Figura 19 – Fluxograma da função *ProcessLinSlave()* executada na LIN *Slave Task*. 47

Figura 20 – Aspecto real do módulo conversor USB-Paralelo UM245R. 50

Figura 21 – Representação gráfica da configuração do módulo UM245R. 51

Figura 22 – Representação da *frame* das mensagens do protocolo implementado. 52

Figura 23 – Fluxograma da função *USBProcessMessage()* executada na IMD e na ECU. 55

Figura 24 – Imagem da janela principal da IMD. 57

Figura 25 – Imagem da janela de configuração da porta COM. 58

Figura 26 – Imagem da janela "*About*" da IMD. 58

Figura 27 – Demonstração do correcto estabelecimento da comunicação através da interface USB. 59

Figura 28 – Demonstração do funcionamento correcto do protocolo LIN e da conexão estabelecida entre todos os módulos. 60

Figura 29 – Demonstração da aquisição do valor de pressão no depósito de combustível (valor adquirido próximo do pretendido). 61

Figura 30 – Demonstração da simulação de falha no protocolo LIN. 61

Lista de Acrónimos

AUTOSAR	*Automotive Open System Architecture*
CAN	*Controller Area Network*
CO	Monóxido de Carbono
CO2	Dióxido de Carbono
CRC	*Cyclic Redundancy Check*
D2B	*Domestic Digital Bus*
ECU	*Electronic Control Unit*
EMC	*Electromagnetic Compatibility*
FC	*Fuel Consumption*
FERR	*Framing Error Bit*
FIFO	*Fist In First Out*
HC	Hidrocarboneto
H2O	Água
ID	*Identifier*
IMD	Interface de Monitorização e Diagnóstico
LIN	*Local Interconnect Network*
Mbb	*Multi Block Byte*
MBT	*Maximum Brake Torque*
MOST	*Media Oriented System Transport*
NOx	Óxido Nitroso
PCB	*Printed Circuit Board*
PDA	*Personal Digital Assistant*

PID *Protected Identifier*

RPM Rotações Por Minuto

SAE *Society for Automotive Engineers*

SCI *Serial Communication Interface*

SENDB *Send Break Character*

TDC *Top Dead Center*

TDMA *Time Division Multiple Access*

TTP *Time Triggered Protocol*

USART *Universal Synchronous Asynchronous Receiver Transmitter*

USB *Universal Serial Bus*

Capítulo 1

1. Introdução e Estado da Arte

1.1. Estrutura do Documento

No primeiro capítulo deste documento é realizado o enquadramento do tema desta dissertação, expondo a motivação que levou à sua execução e os objectivos principais propostos. São ainda apresentadas as áreas fundamentais envolvidas na implementação de uma solução distribuída para o controlo electrónico de um motor de combustão interna, detalhando a sua relevância e contributo no desenvolvimento do trabalho realizado.

No segundo capítulo é apresentado todo o processo de transformação da ECU centralizada na solução distribuída, incluindo uma explicação sobre o sistema e as descrições detalhadas dos vários módulos e software implementados. Deste modo, espera-se o aclaramento do conceito existente nesta implementação distribuída.

A importância e explicação do protocolo de comunicação utilizado na interligação dos vários módulos de controlo são apresentadas no Capítulo 3. Adicionalmente, descreve-se e explica-se detalhadamente a solução de software desenvolvida para a correcta implementação do protocolo.

No Capítulo 4 é apresentada a solução desenvolvida para a comunicação da Interface de Monitorização e Diagnóstico da ECU, a descrição do módulo conversor utilizado, e ainda as características técnicas do protocolo implementado. É igualmente exposta a solução gráfica desenvolvida para a IMD.

Os resultados das implementações expostas, e principalmente a discussão do trabalho desenvolvido são expostos no Capítulo 5. Espera-se desta forma, fornecer um esclarecimento sobre as dificuldades e obstáculos deparados, debater as decisões e soluções implementadas referindo eventuais falhas e erros cometidos.

Por fim, no último capítulo são apresentadas as conclusões do trabalho desenvolvido pretendendo-se mostrar o sucesso e as faltas no cumprimento dos objectivos propostos, justificando-as da melhor forma possível. São ainda expostas algumas sugestões do trabalho futuro a desenvolver no âmbito deste projecto.

1.2. Motivação e Objectivos

O desenvolvimento de uma centralina automóvel com arquitectura distribuída enquadra-se no projecto ICARO (http://icaro.ua.pt). Caracteriza-se por ser uma iniciativa de carácter pluridisciplinar da Universidade de Aveiro, envolvendo áreas como mecânica, design, electrónica, sistemas de informação e informática. Tem como propósitos a participação anual de uma equipa representante da Universidade de Aveiro na prova Europeia da *Shell Eco-marathon* (www.shell.com/eco-marathon), e a potenciação do desenvolvimento e inovação nas áreas associadas. Sendo o objectivo principal da competição a poupança de combustível, e estando em fase de conclusão a construção de um novo veículo, pretende-se agora desenvolver uma nova centralina para o motor de combustão interna baseada num paradigma distribuído.

O novo modelo da centralina com arquitectura distribuída destina-se ao controlo de um motor de combustão interna mono cilíndrico de pequena cilindrada. A solução prevê a implementação de unidades independentes para a execução das várias tarefas de controlo do motor. Desta forma, prevê-se alcançar uma optimização dos tempos de resposta nos elementos principais de controlo do motor, proporcionando liberdade de configuração e ajuste dos mesmos.

É importante referir que o estado de desenvolvimento da centralina havia já sido iniciada em data anterior ao início desta dissertação, estando já instanciados, mas não testados, os vários módulos que a compõem. A construção das placas de circuito impresso dos módulos de controlo estava praticamente terminada. Relativamente ao software, o seu desenvolvimento encontrava-se num estado mais elementar com algumas funções de comunicação provenientes da solução centralizada anterior. Assim, foi a partir do estado em que se encontrava o projecto e usufruindo de todo o conhecimento adquirido ao longo dos anos de participação da equipa na competição, que se deu início a esta dissertação.

Os objectivos propostos foram divididos em duas fases temporais. A fase inicial contemplava a análise do problema, levantamento do estado da arte relativa à tecnologia de controlo electrónico de motores de combustão interna, e análise comparativa do custo/benefício de uma solução distribuída versus centralizada. Pretendia-se ainda, a produção de um documento sobre o estudo efectuado e uma apresentação a realizar no fim desta primeira etapa.

A segunda parte do plano de trabalhos a desenvolver incluía o seguinte conjunto de tarefas:

- Análise do projecto dos módulos electrónicos já desenvolvidos, teste e validação dos mesmos;

- Projecto e desenvolvimento do software de controlo para cada um dos módulos, bem como do protocolo de comunicação entre eles
- Implementação e teste do sistema distribuído de controlo electrónico do motor;
- Redacção da dissertação.

1.3. Sistemas de Tempo-Real

Sistemas de Tempo-Real são sistemas com altas exigências temporais, normalmente associados ao controlo e monitorização de processos físicos, e que por isso devem cumprir critérios de pontualidade e correcção lógica. São designados por tempo-real pois dependem da dinâmica do processo físico envolvido, sendo o sistema obrigado a satisfazer os requisitos temporais próprios desse processo. Implicam, portanto, a gestão eficaz dos recursos e o agendamento organizado das tarefas tendo como objectivo central a execução das mesmas dentro dos limites temporais pretendidos. Neste sentido, é relevante distinguir ritmo de evolução (ou dinâmica) de rapidez, pois um sistema de alta performance (com uma elevada capacidade de resposta) pode não ser classificado como tempo-real se não lhe forem impostas restrições temporais na execução das tarefas[1].

Tome-se como exemplo, um servidor de rede em que o excesso de tráfego provoca atrasos na entrega dos pacotes de informação e consequente perda de qualidade de serviço, mas que no entanto não prejudica catastroficamente o funcionamento da rede (tal como a recepção de um email com alguns segundo de atraso). Apesar de se tratar de um sistema onde se lida com grandes velocidades de comunicação (fala-se portanto de rapidez do sistema), não é necessário garantir a todo o custo um serviço excepcional sem atrasos temporais. Assim, a dinâmica do sistema não impõe que sejam cumpridos prazos rígidos na correcta execução de uma dada tarefa. Como contra-exemplo, considere-se a realização de uma vídeo-conferência em *stream* através desse servidor e que qualquer atraso na transmissão dos dados comprometeria seriamente a comunicação entre os intervenientes. Neste caso, a exigência temporal dessa tarefa implica o cumprimento atempado e sem erros do sistema, sendo a sua resposta crucial para assegurar o correcto funcionamento do mesmo.

Estamos portanto, perante um sistema de tempo-real, quando é possível monitorizar e actuar oportunamente sobre ele de modo a alcançar um comportamento correspondente à dinâmica do próprio sistema, cumprindo restrições temporais e apresentando correcção lógica nos resultados.

1.4. Sistemas Distribuídos

Este é o tema essencial de todo o trabalho. É em torno dos sistemas distribuídos que se focaliza o objectivo desta dissertação, servindo inclusive para a designação da mesma. Deste modo, é de todo conveniente dedicar especial atenção a este tema e compreender de forma clara as características, vantagens, desvantagens e implicações da adopção de uma solução distribuída no âmbito do projecto a desenvolver.

1.4.1. Caracterização

Realizada uma pesquisa inicial é possível encontrar várias definições mais ou menos extensas e/ou completas, sobre o que define um sistema distribuído. A título de exemplo, a citação seguinte descreve um sistema distribuído de forma bastante sintética e ilustrativa: *"A distributed system is several computers doing something together"*[2]. Porém, é da autoria do cientista americano *Leslie Lamport* e remontando a 1987, a mais popular das descrições de um sistema distribuído: *"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable"*[3].

É possível observar que o conceito abordado não é recente, e que das citações apresentadas, um sistema distribuído requer o uso de várias unidades computacionais interligadas entre si através de um meio de comunicação próprio. Deste modo, pode-se resumir a definição de um sistema distribuído a três características fundamentais[2]:

- **Múltiplas unidades computacionais:** um sistema distribuído possui mais que uma unidade física de processamento, mais ou menos complexa (memória local, interfaces de *Input/Output*, etc.);
- **Rede de comunicação**: essencial para estabelecer a comunicação entre os vários elementos computacionais, e deste modo tornar o sistema distribuído;
- **Estado de operação partilhado:** os computadores usufruem da rede de comunicação para partilhar um determinado estado de operação congruente a todo o sistema.

A partir das três propriedades apresentadas, é importante definir um conjunto de aspectos que garantem a correcta implementação de um sistema distribuído, sendo os mais relevantes de seguida enunciados:

- **Autonomia:** os vários computadores devem interagir entre si e ao mesmo tempo garantir independência, isto é, em caso de falha de uma ou mais unidades as restantes devem ser capazes de continuar a operar. A falha individual de uma unidade, idealmente não deve comprometer o sistema;
- **Escalabilidade:** o sistema deve ser capaz de assegurar o correcto funcionamento independentemente do número de nós;
- **Sistema aberto:** deve ser possível adicionar hardware/software sem a necessidade de grandes alterações, ou seja, o sistema deve possuir capacidade de expansão facilitada (*e.g.* definição de protocolos);
- **Falibilidade:** deve-se considerar a hipótese da ocorrência de falhas momentâneas na comunicação entre as várias unidades. Devido às limitações do meio de comunicação, as mensagens podem ser perdidas, conter erros, e até perder-se a conectividade total entre os vários módulos;
- **Segurança:** em certos sistemas a rede de comunicação pode estar exposta a intromissões externas que podem comprometer o correcto funcionamento do mesmo;
- **Custos:** a existência de uma rede de comunicação segura e eficaz entre os vários computadores introduz maiores custos, maior latência na comunicação, e menor velocidade de transmissão do que num sistema centralizado.

1.4.2. Solução Distribuída vs. Solução Centralizada

Neste ponto serão apresentadas as vantagens e desvantagens de uma solução distribuída em comparação directa com uma solução centralizada. Uma vez que o objectivo principal de todo o trabalho consiste na demonstração do desenvolvimento de uma solução distribuída viável para uma centralina automóvel, é primordial destacar as diferenças entre os dois tipos de soluções apresentadas. Deste modo, desde as especificidades técnicas até às contrapartidas financeiras, serão expostas as justificações teóricas para a adopção de uma solução distribuída em detrimento de uma centralizada.

De um modo geral uma solução distribuída é escalável, providencia maior autonomia aos subsistemas devido à sua maior imunidade a falhas, resultando do ponto de vista global, num sistema mais versátil e capaz. Em contrapartida, uma topologia centralizada permite um acesso a dados e recursos de forma equitativa, uma gestão e manutenção mais simplificada, e uma maior facilidade de diagnóstico e alteração do sistema.

Relativamente à segurança e fiabilidade destes dois modelos, não é possível estabelecer uma comparação directa e evidente. Ambos possuem vantagens e desvantagens nestas áreas mas em aspectos diferentes, ou seja, as características de segurança de cada arquitectura não são equiparáveis mas antes adequáveis a cada sistema em concreto. Assim, num modelo centralizado a segurança é analisada de um ponto de vista interno ao sistema pois este existe num ambiente controlado, com uma única entidade reguladora que assegura o bom funcionamento do sistema de forma local. Infere-se que as interferências externas são controladas e praticamente inexistentes. Por outro lado, uma possível falha interna implica o colapso de todo o sistema, embora a detecção e correcção da falha possa eventualmente ser mais simples do que num sistema distribuído. Neste ponto pode-se falar de disponibilidade de um sistema centralizado e na sua capacidade de executar tarefas de forma fiável

No extremo oposto encontram-se os sistemas distribuídos com múltiplos e variados factores de segurança a ter em conta, e dos quais depende o bom funcionamento do sistema. Desde o meio físico de comunicação que envolve um elevado grau de insegurança *per se*, até ao desempenho das diversas unidades que compõem o sistema, revela-se um grande desafio obter a fiabilidade desejada e conciliar os requisitos e comportamentos das várias unidades computacionais. Não obstante, da divisão de competências advém que uma falha de segurança num dado elemento não implica a paralisação de todo o sistema (com a desvantagem, eventualmente, de ser dificultado o diagnóstico e resolução da mesma). Questiona-se então a fiabilidade de um sistema com arquitectura distribuída, podendo-se obter grandes proveitos ou danos dependendo do tipo de implementação seguida. No caso de um sistema em que é necessária intercooperação simultânea de vários computadores para se realizar determinada tarefa, a probabilidade de falha de um qualquer computador é maior que a falha isolada de um dado componente num único computador. No entanto, se encararmos o problema de outra perspectiva este acaba por se tornar numa solução com grandes vantagens para o sistema. Basta para isso projectar um sistema capaz de lidar eficazmente com a eventual falha individual de um computador (salvaguardando funções e variáveis, por exemplo), de modo a que o funcionamento de todo o sistema não dependa exclusivamente de determinadas unidades computacionais. A existência de uma rede de comunicação fiável e segura está inerentemente associada ao cumprimento dos requisitos expostos, representando esta uma parte fulcral no asseguramento dessas funções. Assim, tendo em consideração e respeitando estas características durante a implementação de um sistema distribuído, conseguir-se-á reduzir as probabilidades de falha total e aumentar consideravelmente a fiabilidade e disponibilidade do sistema no cumprimento de tarefas[2].

A uma solução distribuída estão inerentemente associados os custos resultantes da utilização de múltiplos microprocessadores e da implementação de um meio de comunicação eficaz entre eles. Considerando que no final dos anos 70 o preço dos microcontroladores de 8-16 bits situava-se entre $500 e $5000[4], seria financeiramente desencorajador apostar nesse tipo de sistemas. Não obstante já nessa altura se falava em soluções distribuídas como o futuro dos sistemas computacionais, como é demonstrado na citação de seguinte:

"It is my firm belief that the 1980s will be the decade of distributed data processing. The continuing decline in processor and memory cost coupled with lower cost communications, to use a few examples, will hasten the development and widespread use of distributed systems based on micro- and minicomputer technology."[4]

Hoje em dia o seu preço é cerca de 100 vezes menor, reduzindo de forma significativa o obstáculo financeiro para a implementação de um sistema com arquitectura distribuída. De um modo geral, os sistemas centralizados usufruem de um investimento financeiro inicial menor, mas que cresce exponencialmente com o tamanho do sistema. Por sua vez, os sistemas distribuídos possuem um comportamento no crescimento dos custos mais linear, à medida que o sistema se expande[4]. Assim, estes sistemas são uma alternativa financeira e tecnologicamente viável devido à sua inerente distribuição, combinada com a capacidade de especialização no desempenho de tarefas dedicadas. Esta característica permite a execução de operações de tempo-real mais próximas dos sistemas a controlar, declinando a responsabilidade e a carga computacional de um grande e único sistema central de controlo. A sua versatilidade é outra grande vantagem, pois permite o upgrade e a expansão do sistema com uma maior facilidade, incluindo uma redução de custos, tempo, e encargos de implementação e desenvolvimento. Um sistema centralizado, por outro lado, não possui esta capacidade de expansão/actualização resultando num tempo de vida mais reduzido que, quando ultrapassado, implica a substituição total do sistema[4].

No entanto, a vantagem económica de um sistema distribuído pode ser contestada quando a complexidade e requisitos do sistema a implementar são exigentes (e. g. grandes capacidades de processamento, intolerâncias a qualquer tipo de falhas, redes de comunicação com elevadas taxas de comunicação). No caso de um sistema com arquitectura centralizada trata-se sobretudo de um problema a resolver ao nível do software, enquanto num sistema distribuído o desafio se transfere em parte para o hardware e em encontrar a melhor forma de interligar eficazmente as várias unidades de processamento[2].

Considere-se como exemplo a maior solução distribuída existente: a Internet. Uma rede de computadores não é mais que uma solução distribuída com todas as vantagens e desvantagens inerentes. A facilidade de partilha e acesso a informação através de uma rede organizada e amplamente difundida, comodamente acessível a partir de um terminal de baixo custo (e. g. computador, servidor, etc.), com capacidade de expansão teoricamente ilimitada, e uma grande imunidade a falhas (a perda de funcionamento de um terminal não implica a paragem de todo o sistema), são qualidades e vantagens incontestáveis da aplicação de uma solução distribuída em detrimento de um modelo centralizado. No entanto, é indispensável enunciar os possíveis riscos e desvantagens inerentes a este tipo de soluções: aumento da

dificuldade e custos de manutenção, gestão e upgrade; aumento do risco de perda/degradação de informação na rede, e redução da segurança devido à utilização de uma rede de comunicação partilhada (usualmente implementadas em ambientes não controlados).

Finalmente, resta associar a solução distribuída com os sistemas de tempo real. Como já foi descrito anteriormente, um sistema é classificado de tempo real quando apresenta correcção lógica nos resultados e é capaz de satisfazer os requisitos temporais para a disponibilização dos mesmos (respeitando deste modo a dinâmica do sistema). Logo, numa solução distribuída de tempo real a correcção lógica e a exigência temporal são transferidas para os vários nós que compõem o sistema, sobretudo para a rede de comunicação, que desempenha uma função decisiva no funcionamento do sistema. Relativamente à fiabilidade dos vários elementos de um sistema deste tipo, deve ser atribuída à rede de comunicação um grau de importância maior que aos restantes elementos computacionais[5].

1.5. Protocolos de Comunicação no Contexto Automóvel

Neste ponto é apresentada uma explicação sucinta sobre a importância das redes de comunicação no contexto veicular, assim como uma breve apresentação de alguns dos protocolos mais vulgarmente empregues nos automóveis actuais.

Devido à generalizada e crescente utilização da electrónica nos veículos automóveis, sobretudo ao nível dos ECUs (*Electronic Control Units*), foi necessário o desenvolvimento de redes de comunicação capazes de acompanhar a evolução e a expansão dos sistemas de controlo electrónicos implementados nos veículos automóveis modernos[6]. Hoje em dia um automóvel possui dezenas de microprocessadores (em 2002 um veículo da série 7 da marca BMW possuía mais de 65 microprocessadores[7]) que necessitam de estar ligados entre si através de uma ou várias redes de comunicação eficientes, desempenhando uma função crucial no que respeita ao consumo, performance, segurança e conforto dos veículos desenvolvidos. Facilmente se infere que surgiram múltiplas soluções e protocolos de comunicação, desenvolvidos pelos diversos fabricantes e adaptados às crescentes exigências e funcionalidades existentes nos veículos. Não obstante, da necessidade de redução de custos de desenvolvimento e implementação das diversas redes de comunicação, da escassez de ferramentas eficazes que facilitem e simplifiquem o projecto dessas redes, e da falta de normas que auxiliem a interligação dos vários protocolos no ambiente veicular, surgiram várias iniciativas empresariais tendo em vista a regulamentação desta área na indústria automóvel. Consequentemente, foram criados consórcios (LIN *Consortium* e *FlexRay Consortium*) e cooperações (MOST *Cooperation*) formados pelos vários construtores, com o objectivo comum de desenvolver e uniformizar esses protocolos rentável e eficazmente. Ao nível do design, projecto e gestão da arquitectura dos sistemas de comunicação nos veículos foi criado em 2004 o AUTOSAR (*Automotive Open System Architecture*), que providencia a mesma

transparência e organização na arquitectura de sistemas que os protocolos apresentados na área das comunicações intra-veicular[8].

Deste modo, em 1994 a SAE (*Society for Automotive Engineers*) estabeleceu uma classificação para os diversos protocolos de comunicação automóvel. Três classes foram definidas: Classe A, Classe B e Classe C[1], de acordo com a sua largura de banda e funcionalidades atribuídas no contexto de uma rede de comunicação[9]. Na *Figura 1* (obtida em [10]) é possível observar uma representação do crescimento das redes de comunicação no contexto veicular entre 1995 e 2002.

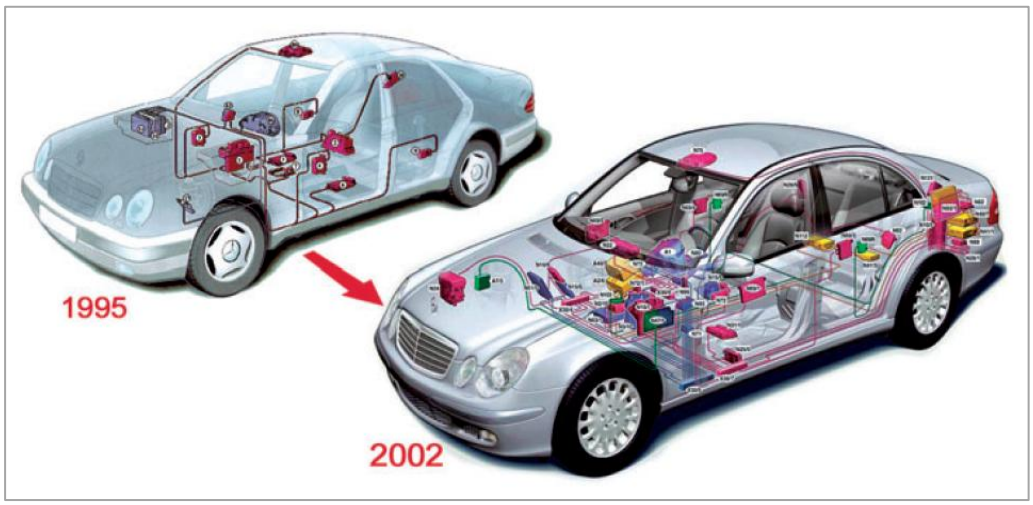


Figura 1 – Evolução das redes de comunicação no contexto automóvel baseado nos modelos W210 (1995) e W211 (2002) da série Classe E da Mercedes-Benz.

Na Classe A estão incluídos protocolos com baixa velocidade de comunicação (inferior a 10Kbit/s) destinados à transmissão de comandos simples de controlo nas áreas de conforto e conveniência (ar condicionado, portas, iluminação, configuração dos bancos, etc.), utilizando tecnologia acessível e de baixo custo. São exemplos de protocolos Classe A o LIN (*Local Interconnect Network*) e o TTP/A (*Time Triggered Protocol for SAE Class A Applications*).

Os protocolos de Classe B são destinados à partilha de mensagens entre diferentes ECUs, nomeadamente informações provenientes de sensores (reduzindo desta forma a redundância de implementação). Deste modo operam a velocidade de transmissão mais elevadas (entre 10Kbit/s e 125Kbit/s) sendo exemplo desta classe o protocolo CAN (*Controller Area Network*).

[1] Existe ainda a Classe D, embora não seja formalmente definida é vulgarmente utilizada para incluir protocolos de comunicação com velocidades de transmissão superiores a 1Mbit/s [9].

Por fim, na Classe C encontram-se os protocolos com velocidades de transmissão superiores a 125Kbit/s e inferiores a 1Mbit/s, utilizados na comunicação a alta velocidade de com características tempo-real. Insere-se nesta categoria o CAN de alta velocidade. Os protocolos com velocidades de comunicação superiores a 1Mbit/s estão incluídos na Classe D, adequados para a transmissão de dados multimédia como o MOST (*Media Oriented System Transport*) e o D2B (*Domestic Digital Bus*), ou aplicações *x-by-wire*[2] que requerem protocolos com tolerância a falhas e previsibilidade como o TTP/C (*Time Triggered Protocol for SAE Class C Applications*), o TT-CAN (*Time Triggered* CAN) e o *FlexRay*[11].

Existem ainda protocolos de comunicação sem fios como o *Bluetooth* e o *ZigBee* que poderão ser utilizados no contexto automóvel num futuro próximo[12], e que devido à sua diminuta expressividade nesse meio não serão abordados. Segue-se a descrição sucinta referente a cada protocolo referido no contexto da sua utilização veicular, à excepção do LIN que será discutido em pormenor no ponto *1.5.1*.

FlexRay

O *FlexRay* foi inicialmente criado pela BMW e Daimler-Chrysler como solução para as futuras gerações de redes de comunicação automóveis, nomeadamente na migração para os sistemas *x-by-wire*. Devido à inexistência nessa altura de um protocolo eficaz na tolerância a falhas, seguro e adequado a esse tipo de aplicações, foi criado o *FlexRay Consortium* constituído por diversas companhias da indústria automóvel. Em 2004 as especificações técnicas do protocolo foram publicadas e actualmente a maioria dos fabricantes faz parte desse consórcio, utilizando o *FlexRay* como solução de uniformização dos diferentes protocolos utilizados.

O *FlexRay* permite taxas de comunicação até 10Mbit/s, projectado para a transmissão de mensagens *time-triggered* e *event-triggered*. Estas características potenciam a sua aplicação no controlo de sistemas de segurança crítica como os *x-by-wire*. Aliando o custo de implementação menor que os protocolos concorrentes da mesma categoria ao suporte declarado da grande parte dos fabricantes automóveis, espera-se transformar o *FlexRay* na solução óptima a adoptar nas aplicações automóveis de segurança crítica e de comunicação a alta velocidade[12].

[2] O termo *x-by-wire* é utilizado na descrição da utilização de sistemas eléctricos e/ou electrónicos em detrimento de mecânicos ou hidráulicos (*e.g. drive-by-wire*, *shift-by-wire*, *steer-by-wire*) [10].

TTP/A e TTP/C

O protocolo de comunicação TTP apresenta duas versões distintas para aplicação no contexto automóvel: o TTP/A e o TTP/C. O TTP/A consiste num protocolo TDMA (*Time Division Multiple Access*) do tipo *master/slave* indicado para aplicações de controlo ao nível do sensor/actuador que não exijam velocidades de transmissão elevadas ou tolerância a falhas. A taxa de transmissão máxima é de 20Kbit/s para um canal de comunicação, permitindo configurações com valores mais elevados. Representa, portanto, uma solução económica e simples para aplicação na área de controlo de sensores e tarefas não críticas. Por outro lado, o TTP/C é um protocolo TDMA totalmente distribuído, orientado à tolerância a falhas, apresentando velocidades de comunicação bastante superiores (máximo de 25Mbit/s). É indicado para aplicações *x-by-wire* devido à sua robustez e alta segurança. No entanto, os elevados custos associados à sua implementação e a baixa flexibilidade tornam este protocolo menos desejável quando comparado, por exemplo, com o *FlexRay*[12].

CAN

Desenvolvido pela *Bosch* na década de oitenta, é actualmente o protocolo de comunicação mais utilizado e disseminado na indústria automóvel. O barramento CAN é enquadrado nas Classes B e C dependendo da velocidade implementada para a utilização pretendida. Este facto torna o CAN um protocolo robusto, versátil e com uma grande variedade de aplicações nas redes de comunicação veiculares, além do seu baixo custo de implementação. É utilizado sobretudo na interligação de ECUs permitindo reduzir significativamente a quantidade de cablagem instalada num veículo. Possui, no entanto, limitações na implementação de sistemas de segurança crítica de alta velocidade (*e.g. x-by-wire*)[11].

TT-CAN

Trata-se da versão *Time-Triggered* do protocolo CAN introduzido em 1999, suportando a transmissão de mensagens *time-triggered* e *event-triggered*. Este facto torna o TT-CAN apto para aplicações *x-by-wire*, sem contudo, proporcionar tolerância a falhas ao mesmo nível dos protocolos da mesma categoria (TTP/C e *FlexRay*). Partilha ainda características base com o CAN, como a topologia da rede, o formato da *frame* e até a velocidade máxima de 1Mbit/s. Actualmente a utilização do TT-CAN em aplicações de segurança crítica que exijam altas taxas

de transmissão é pouco considerada, recorrendo-se a outras soluções mais robustas e eficientes como o *FlexRay*[11].

MOST e D2B

Tratando-se de protocolos orientados para a transmissão de dados multimédia e *infotainment*, a aplicação dos mesmos não se enquadra no âmbito do projecto a desenvolver. Não obstante, o MOST representa o standard consensualmente adoptado pelos construtores, existindo outras soluções como o D2B, utilizado por exemplo, em alguns veículos do fabricante Mercedes-Benz[12].

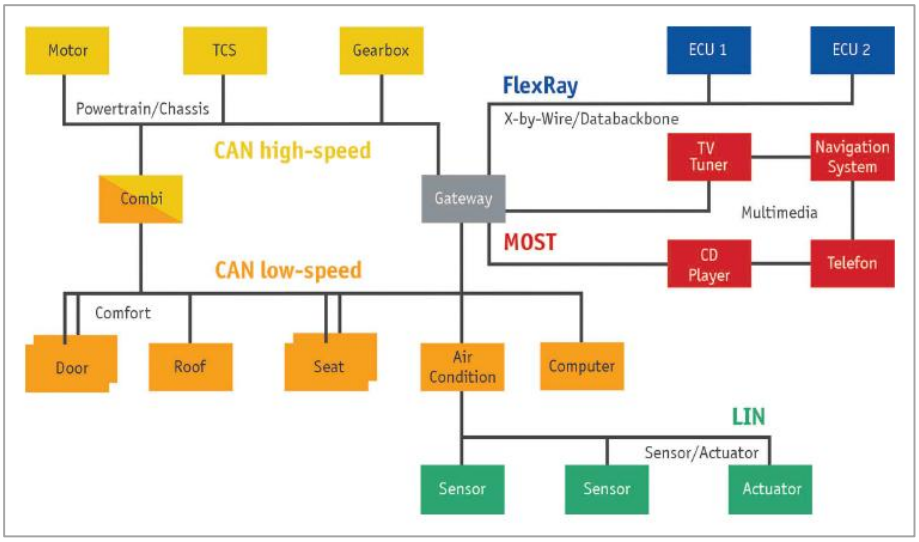


Figura 2 – Representação das diversas redes actualmente existentes em veículos automóveis.

Na *Figura 2* é possível visualizar uma representação da coexistência dos diversos protocolos nas variadas aplicações existentes num veículo automóvel. É facilmente observável que a cada tipo de tarefas encontra-se um determinado protocolo adequado às exigências pretendidas. O exemplo demonstra a utilização do LIN para o nível do sensor/actuador, o CAN de baixa velocidade para as aplicações de conforto (*e.g.* portas, bancos, ar condicionado, etc.), e o CAN de alta velocidade para aplicações ao nível do chassis e dispositivos de locomoção (*e.g.* motor, caixa de velocidades, controlo de tracção, etc.). A um nível superior, o *FlexRay* é utilizado na interligação das várias ECUs e aplicações *x-by-wire*, e em aplicações de multimédia e *infotainment* é empregue o MOST (*e.g.* sistema de navegação, leitor de CD/DVD, etc.).

Depois de apresentados os principais protocolos de comunicação utilizados no contexto automóvel, procede-se à análise e justificação da escolha da solução utilizada no desenvolvimento deste projecto. No âmbito do trabalho a desenvolver interessa considerar como objectivo a adopção de um protocolo de controlo de baixo custo, de fácil desenvolvimento e implementação recorrendo a hardware existente e orientado para o controlo ao nível do actuador/sensor. Além disso, o funcionamento independente dos módulos constituintes da solução distribuída que se pretende desenvolver, não exige velocidades de transmissão muito elevadas para o tipo de mensagens transmitidas.

Deste modo, optou-se pelo LIN pois representa uma alternativa eficiente na implementação de uma rede de comunicação de baixo custo para a interligação de sensores e actuadores[13]. Adicionalmente, permite usufruir da grande facilidade de implementação em qualquer microcontrolador que possua uma Interface de Comunicação Série (SCI), e deste modo aproveitar a disponibilidade imediata de componentes e hardware já existentes[14]. Na *Figura 3* (obtida em[15]) é possível visualizar uma representação ilustrativa da aplicabilidade do protocolo de comunicação LIN no contexto veicular. É indispensável assumir que no respeitante às características de todos os protocolos apresentados e das aplicações alvo associadas a cada um, o LIN possa revelar-se menos adequado ao cumprimento das tarefas a implementar neste projecto. Não obstante, a sua facilidade, simplicidade e comodidade de implementação recorrendo-se a hardware e componentes *off-the-shelf* fazem do LIN um protocolo apetecível e enquadrado nos objectivos do projecto. Naturalmente, destas propriedades advém a vantagem do baixo custo que representa um factor preponderante na decisão da escolha deste protocolo. Pretende-se sobretudo provar desta forma, que o conceito de solução distribuída aplicado a uma centralina automóvel é viável e praticável.

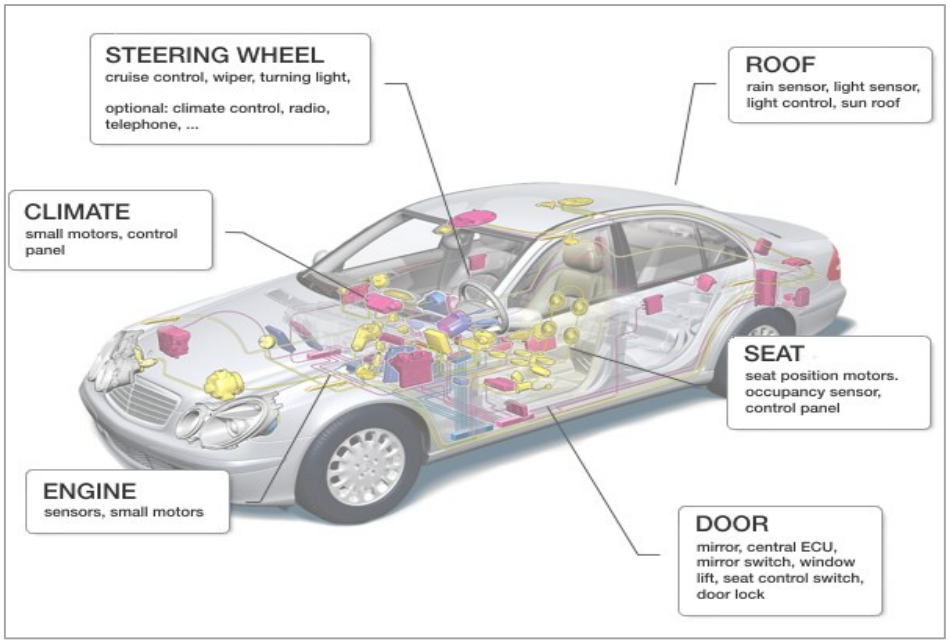


Figura 3 – Representação da aplicabilidade do protocolo LIN no ambiente automóvel.

1.5.1. LIN – Local Interconnect Network

O protocolo de comunicação LIN (*Local Interconnect Network*) é um protocolo de baixo custo essencialmente usado em veículos automóveis, encontrando-se hoje em dia bastante disseminado sobretudo em aplicações de conveniência e conforto (*e.g.* controlo climático, vidros, portas, etc.). Por ser um protocolo de baixa velocidade (máximo 20Kbit/s), foi projectado para ser usado ao nível do sensor/actuador ou seja, para transmissão de informação não crítica que não imponha velocidades de comunicação superiores (*e.g.* CAN). O seu baixo custo de implementação deve-se sobretudo a três características[15]: redução da cablagem na interligação entre os vários nós (apenas usa um fio para comunicação); não requer o uso de um oscilador de cerâmica ou cristal (mais dispendiosos) para sincronização e geração do relógio; utiliza a Interface de Comunicação Série (SCI) incluída na maioria dos microcontroladores, permitindo poupar despesas (e espaço nas Placas de Circuito Impresso – PCB) usualmente associadas à utilização de um controlador de comunicação dedicado[16].

O sucesso e a aprovação deste protocolo deveram-se, em parte, à criação de um consórcio entre os principais fabricantes da indústria automóvel e de semicondutores, cujo objectivo foi criar um standard entre os protocolos desta categoria transversal a todos os construtores; e também à criação de um método de desenvolvimento específico (LIN *Work Flow*) que providenciou a implementação simplificada, eficaz e rentável deste protocolo [8]. Proporcionam, portanto, uma redução na complexidade e a adopção de um único protocolo não proprietário deste tipo, conduzindo a uma diminuição nos custos de produção, desenvolvimento e implementação que na maioria dos casos é factor decisivo na escolha de uma determinada tecnologia.

1.5.1.1. Características Técnicas

A arquitectura deste protocolo é do tipo *Master-Slave*, composta por um nó *master* e um ou mais nós *slave*. Possui uma taxa máxima de transmissão de 20Kbit/s para um máximo de 16 nós físicos em 40 metros de distância, respeitando a compatibilidade electromagnética (EMC) e a sincronização de relógio (condições para as quais se garante o bom funcionamento do bus), sendo a sua camada física baseada na norma ISO 9141. Este protocolo utiliza apenas uma linha de comunicação que faz uso das tensões de alimentação da ECU como referência, e o nível recessivo (nível "1" – quando nenhum nó está a transmitir) é mantido através de uma resistência de *pull-up*.

Na *Figura 4* é possível visualizar uma representação da configuração de uma rede com o protocolo LIN. A sincronização e recuperação de relógio nos *slaves* são feitas sem a utilização de osciladores de cristal ou cerâmicos, e o controlo das comunicações é realizada

recorrendo à implementação das *Slave Tasks* (existente em todos os nós), realizadas por software, e de uma *Master Task* (presente no *master*) que executa periodicamente o LIN *Schedule*. Devido à existência de um e um só *master*, responsável pelo controlo de todo o tráfego, não é necessário hardware adicional para a arbitragem e resolução de colisões ao nível do bit (novamente reduzindo-se custos). A execução da LIN *Schedule* garante a exclusividade de acesso ao *bus*, contendo as frame *slots* associadas a cada *slave*. Assim, é o *master* que através do *broadcast* de uma *frame* contendo uma *Frame Identifier* (ID) – analisada pela *Slave Task* de cada nó *slave* – dá início à comunicação. Esta política de endereçamento baseada em IDs, assegura que apenas o nó correspondente ao ID enviado tomará posse do *bus*. Deste modo, o *slave* responde com uma mensagem composta por uma *frame header* – responsável pela sincronização e contendo o ID do(s) nó(s) destinatário(s) – e uma *frame response* com a informação a ser transmitida.

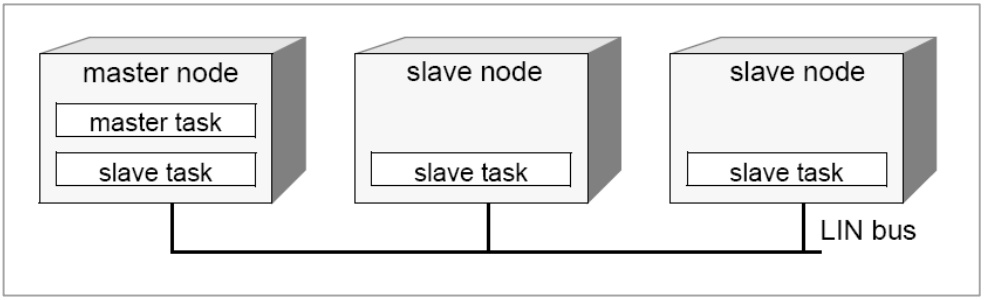


Figura 4 – Esquema representativo da configuração de uma rede LIN.

No protocolo LIN cada mensagem é composta por duas frames: *frame header* e frame response, tal como representado na *Figura 5* (obtida em[15]). A transmissão de cada frame é realizada byte a byte, começando pelo bit menos significativo e com a adição de um *start bit* e um *stop bit* no início e no final de cada byte, respectivamente (formato SCI). Como já foi referido, cada janela é composta por uma *frame header* responsável pela sincronização entre os nós e pela identificação do *slave* emissor e do *slave* receptor (ou dos *slaves* receptores). Devido à utilização de osciladores menos precisos (possibilita a utilização de osciladores com um desvio máximo de 15% do valor nominal[8]), durante a fase de sincronização é necessário que o *master* envie uma pausa de sincronização (*Sync Break*) composta no mínimo por 13 bits dominantes consecutivos, sinalizando os *slaves* para o início de uma transmissão. O *Sync Break* termina com pelo menos um bit recessivo no bus (*Sync Break Delimiter*), indicando que todos os *slaves* estão aptos a receber o byte de sincronização (*Sync Byte*) e a iniciar a transmissão. Apesar do recurso ao uso da pausa de sincronização de pelo menos 14 bits (*Sync Break* + *Sync Break Delimiter*), conjugada com a tolerância significativa na frequência interna de cada nó implicar uma diminuição na taxa de transmissão e introduzir atrasos, esta escolha reflecte-se

nos custos e na simplicidade de implementação deste protocolo. O *header* termina com o envio do *Protected Identifier* (PID) composto por seis bits de identificação e dois bits de paridade para protecção (ímpar e par), possibilitando a identificação de sessenta nós (ID 0 até ao ID 60) sendo os restantes reservados (ID 61 até ao ID 63).

A *frame response* é composta por um máximo de oito bytes de informação e um byte de checksum para detecção de erros na recepção. É possível o uso de dois formatos de checksum: clássico e *extended*. O modo clássico apenas utiliza os bytes de dados para o cálculo da soma, por sua vez no modo *extended* são adicionalmente contabilizados os bytes de PID. Numa situação de erro, em ambos os casos é transmitida uma mensagem de erro.

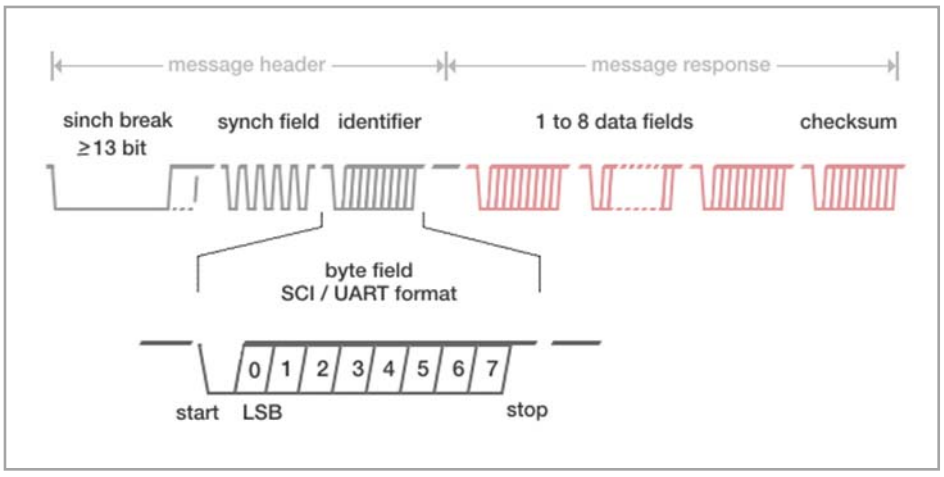


Figura 5 – Exemplo da LIN *frame*, identificado cada campo descrito anteriormente.

Durante a transmissão das frames, pode ser necessária a introdução de espaços entre cada SCI byte (*Interbyte Spaces*) e entre o *header* e o response (*Response Space*), provocado atrasos e prolongando o envio da *frame response* até um máximo de 40% (de forma semelhante ocorre com o *header* devido ao *Sync Break*). Assim, é de todo o interesse considerar este atraso no dimensionamento do LIN *Schedule*.

As especificações deste protocolo contemplam a existência de mensagens de diagnóstico, bem como funções de *Status Management* e *Network Management*. A função de *Status Management* é utilizada para sinalizar o *master* da ocorrência de erros (paridade ou *checksum*), no entanto sem disponibilizar mais informação sobre o tipo de erro ocorrido. Por sua vez, a função de *Network Management* é principalmente usada no controlo da passagem de estado dos *slaves* do modo *Operational* para o modo *Sleep*, e vice-versa. Os *slaves* possuem a capacidade de entrarem em *Sleep Mode* se não for verificada actividade no *bus* durante 4 segundos consecutivos, e retornam ao *Operational Mode* (passando pelo estado de Inicialização) através da recepção de um *Wake Up Signal* e de um *header* válido.

1.6. Controlo Electrónico Automóvel

Actualmente, é incontestável a dependência da indústria automóvel na electrónica e no controlo electrónico. A evolução exponencial desta área tecnológica impulsionou o desenvolvimento e crescimento do ramo veicular tal como o conhecemos hoje. Automóveis mais seguros e confortáveis, mais eficazes, mais ecológicos e amigos do ambiente, mais económicos e tecnológicos, são o resultado de uma simbiose de sucesso estabelecida há anos atrás. O surgimento de problemas na área automóvel que a mecânica e electromecânica convencional não eram capazes de resolver, e mais tarde a imposição de regras governamentais rígidas para o controlo da emissão de gases (que está directamente relacionada com o consumo de combustível)[9], constituíram os principais factores para a adopção da electrónica como a solução mais adequada.

No entanto, foi com o aparecimento dos microcontroladores que se tornou óbvia a relevância e a mais-valia da electrónica na indústria automóvel. Actualmente, este ramo industrial é responsável pelo consumo de cerca de 5% de todo o silício produzido mundialmente, com tendência a aumentar[7]. A evolução tecnológica conjugada com a descida gradual dos custos dos microcontroladores, permitiu alargar a sua aplicabilidade desde o controlo de motores até ao controlo de tracção, transmissão, travagem, suspensão, direcção, iluminação, conforto, etc. Deste modo, considerando o contexto do trabalho a desenvolver, é no controlo electrónico de motores de combustão interna que centraremos a nossa atenção ao longo desta dissertação. Nos pontos seguintes é apresentada uma breve abordagem a esse tema.

1.6.1. Controlo Electrónico de Motores de Combustão Interna

O controlo electrónico de motores de combustão interna pode ser decomposto em três partes: leitura e monitorização dos vários sensores do motor; unidade electrónica de controlo (ECU) que analisa e processa a informação obtida dos sensores e que de acordo com algoritmos e/ou tabelas efectua o controlo do sistema; dispositivos actuadores que executam uma função controlada pela ECU em resposta à informação obtida nos sensores[9]. A utilização da electrónica no controlo de motores de combustão interna permite alcançar funcionalidades, vantagens e cumprir objectivos que de outra forma seriam impraticáveis. Desde a inerente precisão e versatilidade associada à electrónica, à possibilidade de simulação e diagnóstico de avarias (permitindo poupar tempo e custos), e respondendo às imposições ambientais, ecológicas e económicas, o controlo electrónico é actualmente indispensável e fundamental na indústria automóvel global.

O controlo e monitorização de um motor de combustão interna envolve a utilização de vários sensores e actuadores para medir e actuar sobre um vasto conjunto de variáveis. De um modo geral os aspectos mais importantes e que requerem maior precisão e exigência no seu controlo são as emissões de gases de escape e o consumo de combustível, considerando evidentemente, o desempenho e a qualidade de condução desejadas. No contexto do projecto a desenvolver todo o trabalho e investigação é direccionado para o controlo e redução do consumo de combustível (objectivo principal da competição *Shell Eco-Marathon*), estando este directamente relacionado com a emissão dos gases de escape. Assim, são estes dois aspectos que interessará aprofundar de seguida e em torno dos quais se desenvolverá o trabalho.

1.6.1.1. Emissões de Escape, Consumo de Combustível e Força Produzida

As emissões de escape consistem num conjunto de produtos resultantes da reacção de combustão da mistura ar-combustível, que são expelidos para a atmosfera durante o funcionamento do motor. Em condições ideais de combustão a reacção térmica combinaria o combustível (idealmente constituído por hidrocarbonetos) com o oxigénio do ar resultando na formação de dióxido de carbono (CO2) e água (H2O). No entanto, na prática, é impossível obter uma combustão perfeita levando à formação de outros compostos como o monóxido de carbono (CO), os óxidos nitrosos (NOx) e os hidrocarbonetos (HC)[9]. É a quantidade emitida deste tipo de compostos poluentes que necessita de medição por parte dos sensores, e posterior controlo e regulação através da ECU e respectivos actuadores do sistema.

No processo de controlo das emissões dos gases de escape de um motor, vários aspectos são tidos em conta sobretudo a relação da mistura ar-combustível injectada e o momento da ignição (*ignition timing*). São estes os dois aspectos mais decisivos e importantes para a emissão dos gases de escape. Relativamente à mistura ar-combustível esta deve respeitar uma relação entre a quantidade de ar e de combustível ideal para ocorrer a ignição e combustão óptimas (quando todo o combustível é consumido), designada de razão estequiométrica. Nos motores de explosão o valor dessa proporção é de 14.7:1, ou seja, 14.7Kg de ar para 1Kg de combustível. Geralmente é expressa em termos de um factor de excesso de ar (comparativamente à razão estequiométrica) representando o seu desvio a partir do valor ideal, sendo conhecida como lambda (λ). Assim, resulta que para uma mistura ar-combustível com excesso de ar (comummente denominada de mistura pobre) o valor de lambda é maior que 1, e que para uma mistura rica (insuficiência de ar) lambda é menor que 1.

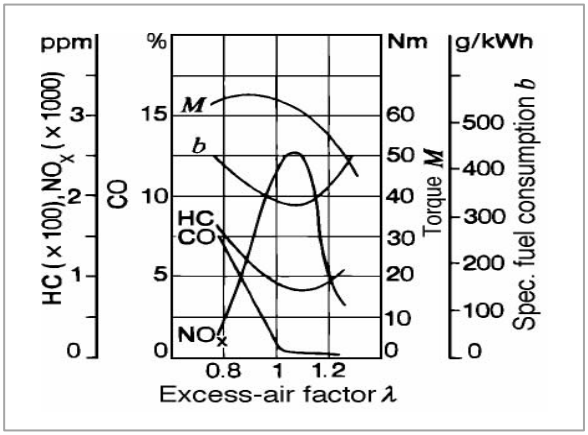


Figura 6 – Efeito de lambda nas emissões dos gases de escape, torque produzido e consumo de combustível.

A mistura ar-combustível condiciona fortemente as emissões de gases de escape e a sua influência reflecte-se na quantidade emitida de cada um dos compostos acima referidos. Também influência significativamente a força produzida pelo motor (torque) e ainda o consumo de combustível. Na *Figura 6* (obtida em [17]) é possível observar essas relações e facilmente compreender a importância dada à mistura ar-combustível e às suas correctas proporções.

Assim, para misturas pobres ($\lambda > 1$) observam-se emissões quase nulas de CO, redução da produção de NOx, e um ligeiro aumento da formação de HC. Regista-se ainda uma redução gradual no torque produzido e o mínimo do consumo de combustível para misturas pouco pobres ($\lambda \cong 1.1$), em contraste com um aumento gradual para valores de lambda maiores. Por outro lado, para misturas ricas ($\lambda < 1$) registam-se emissões elevadas e semelhantes de CO e HC, redução de NOx, e aumento significativo do consumo de combustível. Para misturas pouco ricas ($\lambda \cong 0.9$) o torque é máximo, decrescendo com a diminuição do valor de lambda. Importa ainda mencionar que para a razão estequiométrica ($\lambda = 1$) existe um equilíbrio entre a força produzida, o combustível consumido e a quantidade de gases produzidos, à excepção dos óxidos nitrosos (NOx) que se encontram perto do seu valor máximo de produção.

O momento da ignição (*ignition timing*) constitui outro factor condicionante da emissão de gases de escape e dispêndio de combustível. O momento em que acontece a ignição depende da posição angular da cambota (e consequentemente do êmbolo[3]) em relação ao *Top Dead Center* (TDC – ponto em que o êmbolo atinge o máximo da sua excursão), e da

[3] Peça móvel que se desloca dentro do cilindro e que transmite a força de pressão gerada no interior da câmara de combustão para a cambota. É vulgarmente designado por pistão [18].

combinação ar-combustível utilizada, obrigando a um controlo temporal extremamente exigente e preciso por parte da ECU.

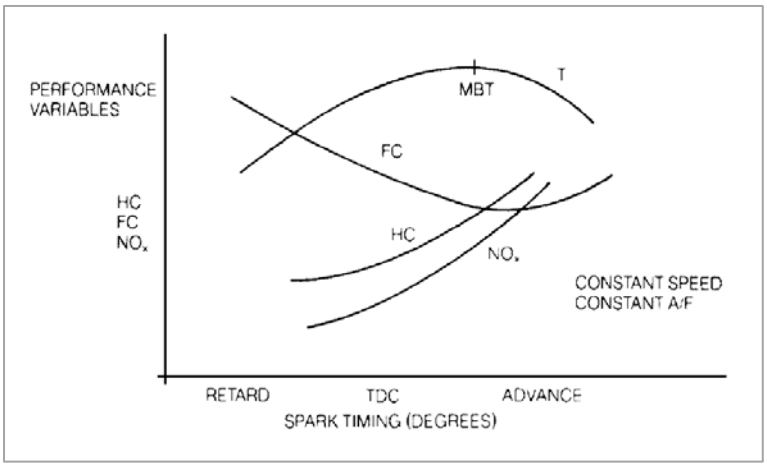


Figura 7 – Efeito do *timing* de ignição nas emissões dos gases de escape, torque e consumo de combustível.

Relativamente às emissões de escape o *timing* de ignição praticamente não influencia a emissão de CO, sendo esta apenas dependente da mistura ar-combustível. No entanto, com o avanço do momento de ignição a formação de HC aumenta, assim como a quantidade de NOx. Na *Figura 7* (obtida em [18]) é possível visualizar a interdependência entre o *timing* da ignição e a quantidade de gases produzidos, o torque gerado (T) e o combustível consumido (FC – *Fuel Consumption*), para uma rotação e mistura ar-combustível constantes. Repare-se que o máximo torque gerado (MBT – *Maximum Brake Torque*) e o mínimo consumo de combustível são atingidos quase no mesmo momento de ignição.

A partir de uma análise conjunta às duas figuras apresentadas, é possível obter algumas conclusões relativamente ao compromisso entre as emissões de escape e o consumo de combustível. Da *Figura 6* verifica-se que para atingir o mínimo de consumo de combustível é necessário utilizar uma mistura ligeiramente pobre. Nestas condições é necessário adiantar o *timing* de ignição para compensar a diminuição da velocidade da combustão devido à redução de combustível na mistura[9]. Não obstante, da *Figura 7* depreende-se que adiantando o timing de ignição origina um aumento acentuado da produção de HC e NOx.

Assim, é com o objectivo principal de gerir e optimizar este compromisso entre as emissões dos gases de escape e o consumo de combustível, manipulando a mistura ar-combustível e controlando o momento de ignição, que o controlo electrónico de motores de combustão interna se revela essencial e insubstituível.

1.7. Motor do Ícaro – Controlo Electrónico

Com o objectivo de esclarecer o princípio de funcionamento e controlo, neste ponto abordar-se-á a descrição do funcionamento do motor que é usado no veículo e dos diferentes tipos de sensores e actuadores aplicados.

1.7.1. Princípio de Funcionamento

O motor do veículo do projecto Ícaro é um motor de explosão mono cilíndrico de funcionamento a quatro tempos. É de todo conveniente efectuar uma breve descrição funcional sobre este tipo de motores, uma vez que para o controlo electrónico dos mesmos é imprescindível o conhecimento da sua dinâmica de operação.

A designação de "quatro tempos" resulta da caracterização das fases necessárias à realização de um ciclo completo, que se desenrola a cada duas rotações do motor[19]. Deste modo, definem-se quatro fases (ou tempos): Admissão, Compressão, Ignição e Exaustão, durante as quais o motor produz duas rotações completas. Na *Figura 8* (obtida em *wikipedia.org*) além de se observarem as ilustrações correspondentes às quatro fases descritas, estão ainda representadas duas suplementares – Estado Inicial (1) e Produção de Força (5) – com o objectivo de auxiliar na compreensão do seu modo de operação.

Considerando a imagem correspondente ao número dois da *Figura 8*, observa-se a descida do êmbolo simultaneamente com a abertura da válvula de admissão, forçando a entrada da mistura ar-combustível para a câmara de combustão: está-se na fase de Admissão. De seguida, na imagem correspondente ao número três observa-se a subida do êmbolo ao mesmo tempo que é comprimida a mistura ar-combustível, correspondendo à fase de Compressão. Nesta altura todas as válvulas estão fechadas e o movimento de ascensão do êmbolo é proporcionado através do momento angular armazenado no volante[4] do motor (ou pelo accionamento do motor de arranque). No ponto quatro o êmbolo atinge o máximo da sua excursão, encontrando-se no ponto mais afastado da cambota (TDC) e iniciando-se a ignição e consequente combustão da mistura. Trata-se da terceira fase do ciclo, a Ignição. É nesta etapa que se produz a força rotacional (ou torque) do motor criada pela combustão da mistura ar-combustível (aumentando a temperatura e consequentemente a pressão na câmara de combustão), provocando a descida do êmbolo até ao ponto mais próximo da cambota (ilustração número 5 da figura). Por fim, na última etapa correspondente ao número 6 da *Figura 8* dá-se o escape dos gases resultantes da combustão da mistura, através da abertura da

[4] Peça rotativa de grande massa ligada à cambota, responsável pelo armazenamento de energia rotacional, proporcionando estabilidade no movimento de rotação do motor. [18]

válvula de exaustão e subida do êmbolo novamente em direcção ao TDC, reiniciando todo o processo e dando origem a um novo ciclo.

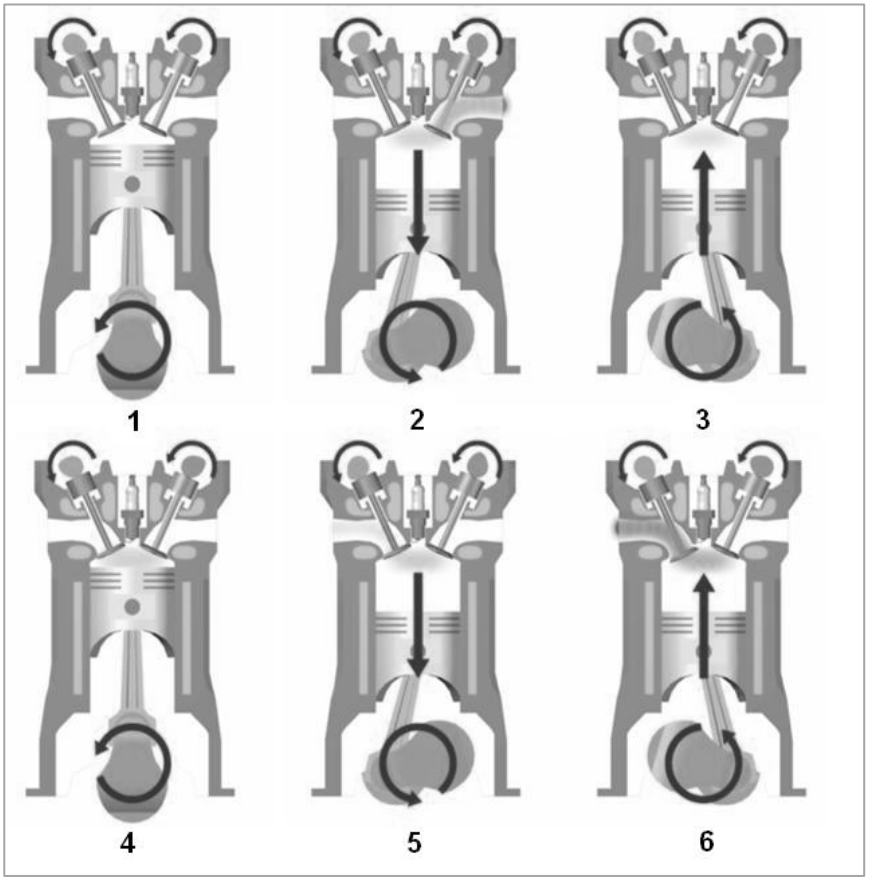


Figura 8 – Esquema ilustrativo das fases de um ciclo completo num motor de explosão a quatro tempos.

Legenda: **1.** Estado Inicial, **2.** Admissão, **3.** Compressão, **4.** Ignição, **5.** Produção de Força, **6.** Exaustão.

Depois da explicação sobre o modo de operação dos motores a quatro tempos, torna-se necessário elucidar sobre as variáveis passíveis de controlo electrónico. Assim, e como já foi referido em *1.6*, importa controlar a quantidade de combustível injectado de forma a obter a mistura ar-combustível desejada, e o timing da ignição. Interessa também monitorizar o factor lambda na saída de escape, a rotação instantânea do motor, a pressão do combustível no depósito, e ainda diversos indicadores como a temperatura externa e do óleo no motor.

1.7.2. Sensores e Actuadores Utilizados

Na continuidade do tema anterior respeitante ao funcionamento do motor usado e conjugando com o controlo electrónico do mesmo, o controlo e monitorização das múltiplas variáveis serão agora aprofundados através da apresentação dos vários sensores e actuadores aplicados no motor. Na lista seguinte estão discriminados os múltiplos sensores instalados no motor do Ícaro:

- **Sensor de Lambda:** utilizado para a medição do factor lambda nos gases de escape;
- **Sensores de Temperatura:** usados para a medição da temperatura da cabeça e do óleo do motor, assim como do ar no colector de admissão;
- **Sensor Angular de Indução:** aplicado para monitorizar a rotação do motor e conhecer a posição do êmbolo;
- **Sensor de Pressão de Ar:** utilizado na medição da pressão do ar no colector de admissão;
- **Sensor de Pressão Atmosférica:** usado na medição da pressão atmosférica;
- **Sensor de Pressão do Combustível:** responsável pela medição da pressão no tanque de combustível.

A utilidade individual de cada sensor enunciado está implícita na sua descrição. Mais importante é referir a sua função no contexto da dinâmica de funcionamento do motor. Assim, os mais relevantes são: Sensor Angular de Indução, Sensor de Pressão de Ar, Sensor de Lambda e Sensor de Pressão de Combustível.

O Sensor Angular de Indução permite conhecer a posição rotacional do motor através da colocação de vinte e quatro pontos de referência num volante concêntrico à cambota do motor (resultando numa resolução de 15º). Na realidade estão instalados apenas vinte e três pontos, uma vez que a omissão propositada do 24.º marcador é coincidente com o TDC permitindo deste modo saber quando este é atingido. Não obstante, como se trata de um motor a quatro tempos, por cada ciclo completo o êmbolo passa pelo TDC duas vezes (ilustrações número 1 e 4 da *Figura 8*). Torna-se necessário recorrer a um sensor adicional e cruzar a informação obtida de modo a saber com precisão em que fase o motor se encontra, requisito essencial para o controlo do mesmo. Essa informação suplementar obtém-se a partir do Sensor de Pressão de Ar colocado no colector de admissão. Sabe-se que durante a fase de

Admissão o êmbolo desce forçando a entrada da mistura ar-combustível para a câmara de combustão, permitindo dessa forma detectar a diferença de pressão ocorrida durante o ciclo de Admissão. Neste momento já se possuem todos os dados fundamentais para conhecer a posição do êmbolo dentro do motor, assim como a fase exacta em que se encontra o motor.

Relativamente ao Sensor de Lambda, a utilidade deste consiste na análise dos gases de escape de modo a permitir efectuar correcções tanto na mistura ar-combustível utilizada como na temporização da ignição. Por sua vez, o Sensor de Pressão do Combustível é utilizado na medição da pressão no interior do depósito e juntamente com o actuador respectivo manter uma pressão desejada constante.

Além de todos os sensores acima descritos, são utilizados cinco formas de actuação de modo a possibilitar o controlo total do motor:

- **Servo da Borboleta:** responsável pelo controlo e regulação da quantidade de ar na mistura ar-combustível, permitindo acelerar/desacelerar o motor;
- **Válvula de Pressão do Combustível:** permite regular a pressão no depósito do combustível;
- **Injector:** responsável pela injecção do combustível e regulação da sua quantidade;
- **Vela de Ignição:** elemento responsável pela ignição e consequente despoletar da combustão;
- **Motor de Arranque:** possui a função de accionamento do motor de combustão interna na fase de arranque inicial, até à sua operação autónoma.

Relativamente aos actuadores, e repetindo o que já foi referido anteriormente, a relevância recai no controlo do timing de ignição e da mistura ar-combustível. Assim, a Vela de Ignição, o Injector e o Servo da Borboleta representam os elementos cruciais para a operação do motor. A Válvula de Pressão do Combustível permite regular a pressão a que se encontra o combustível no depósito e que é disponibilizada para o Injector. A pressão do combustível, em conjugação com o tempo de abertura do injector, permite controlar a quantidade de combustível injectado em cada ciclo do motor.

Capítulo 2

2. Desenvolvimento da Solução Distribuída

2.1. Migração do Sistema Centralizado para o Sistema Distribuído

No ponto *1.7.2* foram discriminados os vários sensores e actuadores aplicados no motor, assim como explicados os seus propósitos. Para além desse conjunto de sensores serão utilizados quatro módulos computacionais, independentemente dedicados à monitorização, gestão e controlo dos diversos sensores e actuadores do motor. Todos eles se encontram interligados através do protocolo LIN[5], partilhando informação e materializando o controlo do sistema em arquitectura distribuída.

Assim, com o objectivo de dissociar as tarefas temporalmente mais exigentes, foram atribuídas responsabilidades distintas aos diferentes módulos, os quais foram desenvolvidos de acordo com essas especificidades. São eles:

- **Módulo de Controlo Central (*Master*):** responsável pelo controlo geral do motor, assumindo a função de *master* no protocolo LIN e efectuando a comunicação com o resto do sistema[6] através do protocolo CAN. Além disso, possui também uma interface USB (*Universal Serial Bus*) de comunicação com o exterior para efeitos de diagnóstico e afinação;

- **Módulo de Controlo de Ignição:** dedicado exclusivamente ao controlo da ignição e da temporização associada;

- **Módulo de Controlo de Injecção:** módulo dedicado ao controlo da injecção de combustível no motor, responsável pela sincronização com o módulo anterior e controlo do motor de arranque;

[5] Além do LIN, está incluída uma linha exclusiva de sincronização entre dois módulos que é abordada em *2.3.6*.

[6] Entenda-se como resto do sistema o volante do veículo que permite o controlo do motor, incorporando um PDA (*Personal Digital Assistent*) para monitorização e controlo de várias operações durante a condução.

- **Módulo de Controlo de Pressão do Combustível:** responsável pela medição e regulação da pressão no tanque de combustível.

Na *Figura 9* da página seguinte é possível observar o esquema representativo do motor do Ícaro, dos múltiplos sensores e actuadores usados, dos quatro módulos e os respectivos *buses* de comunicação. Resume-se pois, graficamente, tudo o que foi descrito e explicado neste capítulo até agora.

Relativamente ao controlo electrónico do motor, este poderá ser efectuado de duas formas: com valores pré-calculados, medidos e posteriormente armazenados sob a forma de tabelas, recorrendo-se aos sensores e actuadores de forma a seleccionar a configuração mais adequada a cada instante para a injecção e ignição (sendo este o método mais usual); ou com algoritmos que efectuam o controlo *on the fly* (dinamicamente) da injecção e ignição, incluindo-se nesta opção a medição das concentrações dos gases de escape (através do sensor de lambda) e a quantidade de ar utilizado na admissão (através de um sensor de massa de ar). Ambos os tipos de controlo obrigam à execução de ensaios de bancada (*e.g.* utilizando um dinamómetro[7]) e de campo (usando o próprio veículo em pista), de forma a obter dados essenciais para a elaboração dos algoritmos e/ou tabelas de controlo.

[7] Equipamento usado para medir o torque e a potência produzidos por um motor nas suas gamas funcionais de rotação e carga [18], e deste modo conhecer o seu comportamento operacional.

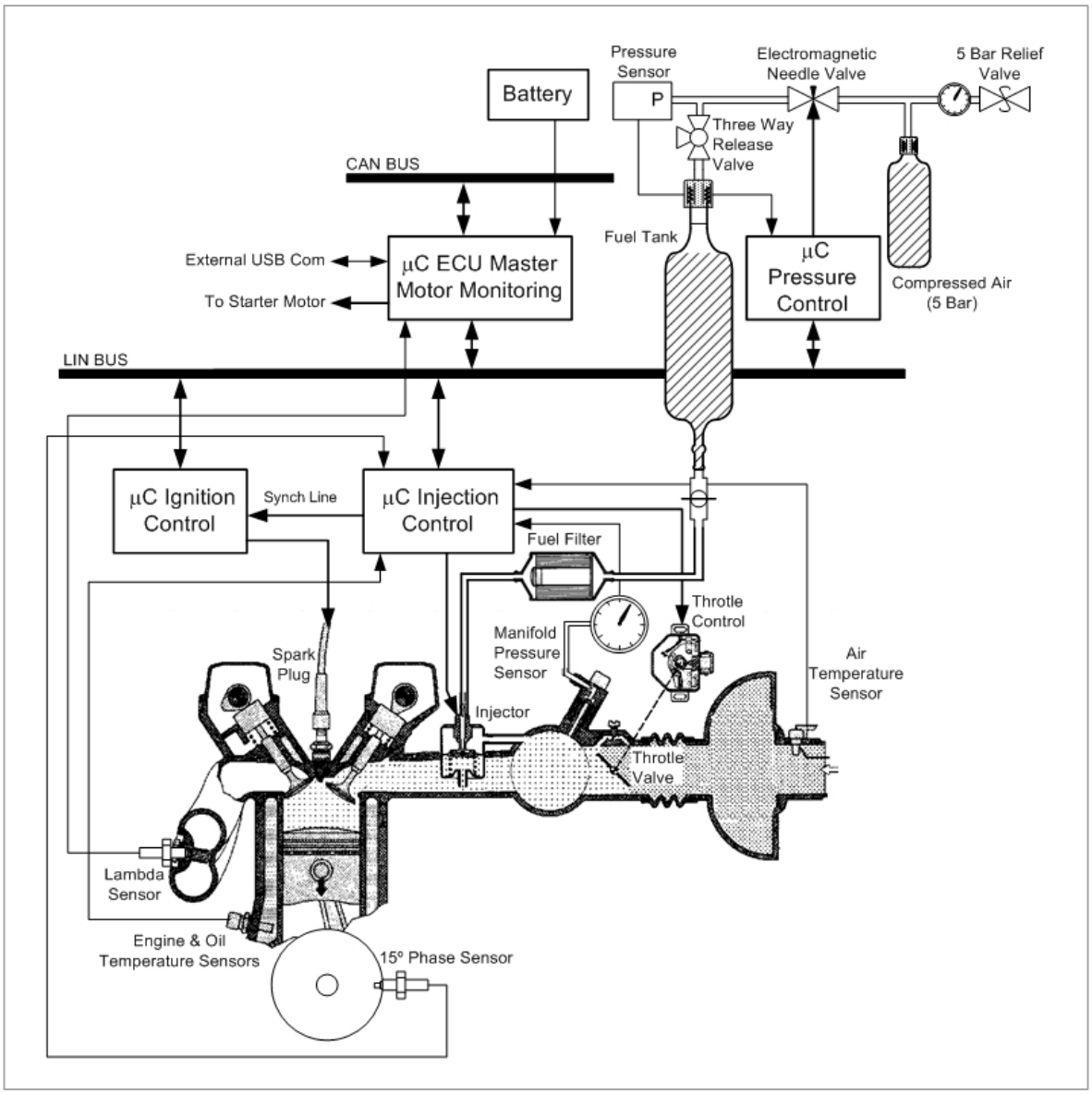


Figura 9 – Representação esquemática do motor, sensores, actuadores e ECUs previstos para o controlo electrónico do mesmo.

É indispensável mencionar que todo o software contido nos módulos foi implementado recorrendo à linguagem de programação C em conjunto com o ambiente de desenvolvimento MPLAB IDE v8.10 da *Microchip*©. Utilizaram-se igualmente, compiladores de software necessários para os microcontroladores instalados nos módulos.

2.1.1. Módulo de Controlo Central (*Master*)

O Módulo de Controlo Central, ou *Master*, desempenha uma tarefa coordenadora e central na operacionalidade de todo o sistema. Na *Figura 10* é possível observar o aspecto real do Módulo de Controlo Central, constituído por um conjunto de blocos funcionais que se podem observar e descrever sucintamente da seguinte forma:

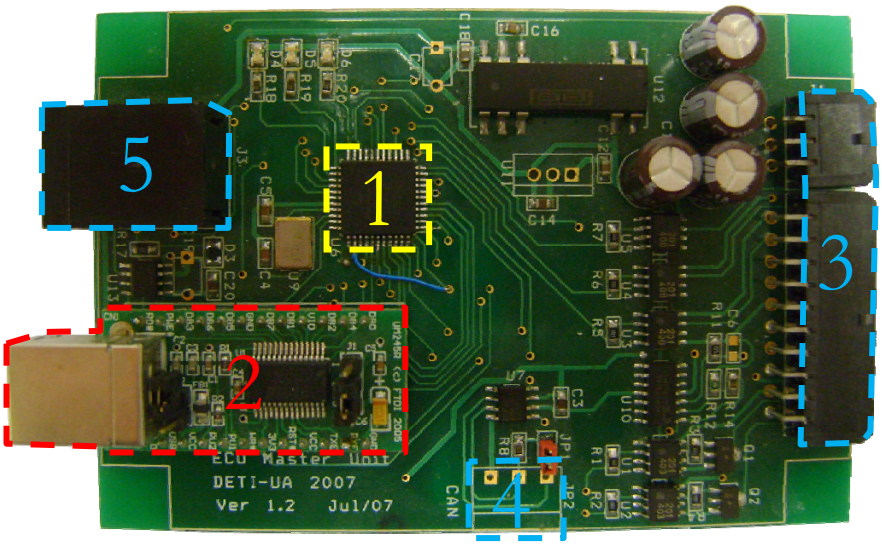


Figura 10 – Aspecto real do Módulo de Controlo Central com as unidades descritas assinaladas.

Legenda: **1.** Microcontrolador; **2.** Módulo conversor USB-Paralelo; **3.** Interface de I/O (LIN, sensores, actuadores e alimentação); **4.** Interface de comunicação CAN; **5.** Interface de programação do microcontrolador.

- **Microcontrolador PIC18F4585:** representa a unidade reprogramável de processamento do módulo, sendo a responsável pela execução de todas as operações de controlo, monitorização e comunicação a si atribuídas;
- **Comunicação LIN:** desempenha a função de *Master Node*, estando por isso implementada a *Master Task* que executa periodicamente a LIN *Schedule*. É responsável, pelo agendamento e controlo das comunicações entre todos os nós existentes no barramento LIN, actuando portanto, como agente despoletador das mesmas;

- **Comunicação CAN:** o barramento CAN está implementado neste módulo para permitir a comunicação com o resto dos sistemas electrónicos do veículo, ou seja, representa a interface através da qual a centralina se torna parte integrante do veículo. Deste modo, ao barramento CAN é atribuída a responsabilidade de recepção de ordens provenientes do *cockpit* (*e.g.* aceleração, arranque ou paragem do motor, etc.), assim como a transmissão de informação relevante para o piloto (*e.g.* estado do motor, consumo de combustível, temperatura do motor, RPM, entre outras);

- **Interface USB:** representa a interface de comunicação com o exterior para efeitos de diagnóstico, *debug* e para efectuar ajustes de afinação no motor. Juntamente com a interface gráfica desenvolvida (apresentada em *4.2*), permitirá monitorizar em tempo real os diversos parâmetros (*e.g.* temperaturas, pressões, RPM, etc.), ajustar novos valores (*e.g.* pressão do combustível) e actualizar as tabelas de configuração do motor;

- **Medição de Lambda:** o factor lambda é medido neste módulo e periodicamente partilhado com os módulos que solicitem essa informação.

As principais funcionalidades do Módulo de Controlo Central estão assim descritas. Além dessas, possui características secundárias adicionais como o controlo do *buzzer*, a medição da voltagem na bateria de alimentação e ainda a medição da velocidade instantânea do veículo e do tempo de prova. Assim, este módulo assume-se fundamentalmente como uma unidade intermédia de comunicação entre a centralina como solução distribuída e o restante sistema electrónico do veículo, uma vez que todas as interfaces de comunicação estão a si conectadas.

2.1.2. Módulo de Controlo da Pressão do Combustível

O Módulo de Controlo da Pressão do Combustível representa a unidade com a menor carga computacional associada. Uma vez que as únicas tarefas a desenvolver por este módulo se resumem à medição e controlo da pressão no interior do depósito de combustível, como se observa na *Figura 9*, a exigência de processamento é menor relativamente aos módulos restantes. No entanto, não é menos necessária a descrição em detalhe dos seus blocos funcionais e a indicação dos mesmos na *Figura 11*:

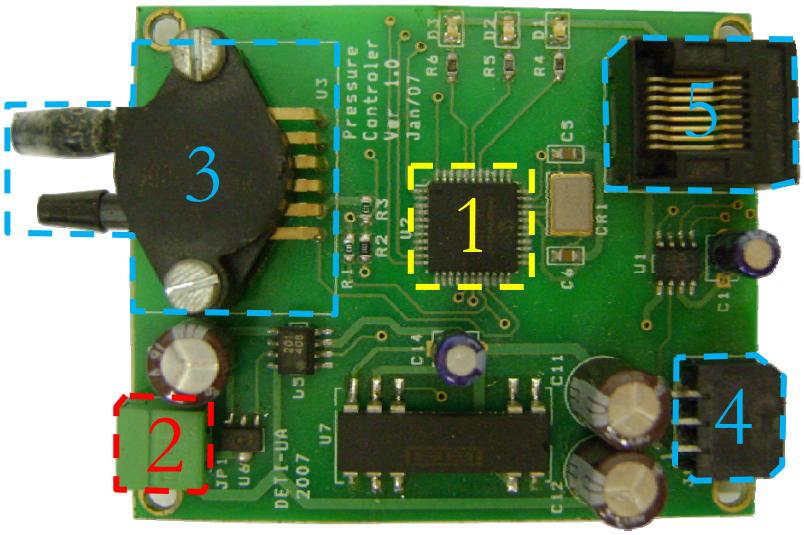


Figura 11 - Aspecto real do Módulo de Controlo Central com as unidades descritas assinaladas.

Legenda: **1.** Microcontrolador; **2.** Interface de ligação à válvula de ar electrónica; **3.** Sensor de medição de pressão diferencial; **4.** Interface de comunicação LIN e alimentação; **5.** Interface de programação do microcontrolador.

- **Microcontrolador PIC18F458:** constitui a única unidade computacional reprogramável do módulo, e por isso, desempenha essencialmente as tarefas de controlo e comunicação a si designadas;

- **Medição de Pressão:** recorrendo a um sensor de medição de pressão relativa do ar (neste caso em relação à pressão atmosférica), é obtido periodicamente o valor da pressão a que se encontra o tanque de combustível;

- **Controlo de Pressão:** através da actuação numa válvula de ar electrónica controla-se a pressão no interior do depósito, aumentando-a de acordo com as especificações desejadas (a diminuição da pressão é efectuada manualmente);

- **Comunicação LIN:** estando o módulo ligado ao barramento LIN, neste é executado uma *Slave Task* de modo a assegurar a sua correcta conectividade ao mesmo. Deste modo obtém-se um estado partilhado entre os módulos, característica inerente de uma solução distribuída. A comunicação LIN torna possível a partilha de informação local (periódica ou *on demand*), assim como a alteração do valor da pressão no interior do tanque de combustível.

2.1.3. Módulo de Controlo da Injecção

Esta é indubitavelmente, a unidade mais complexa e com a maior carga computacional de todo o sistema referente ao controlo electrónico do motor. No Módulo de Controlo da Injecção existem duas unidades dedicadas de processamento – reflexo da complexidade e exigências atribuídas – assim como uma unidade de armazenamento de dados complementar, estando-lhe ainda atribuídas as tarefas de monitorização e controlo de diversos parâmetros.

De seguida, na *Figura 12* e *Figura 13* apresenta-se o aspecto real dos lados anterior e posterior do módulo e explicam-se as funcionalidades principais das unidades destacadas:

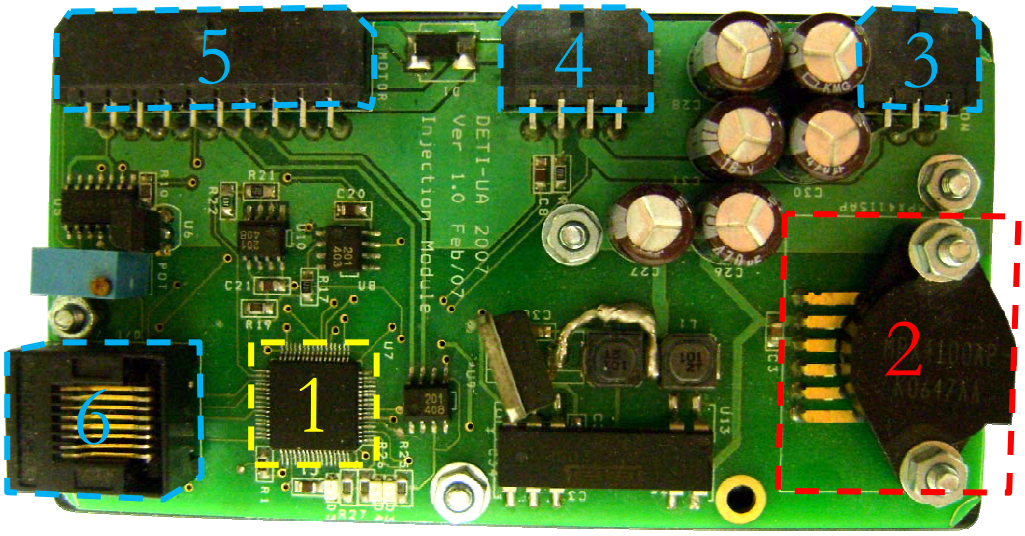


Figura 12 - Aspecto real anterior do Módulo de Controlo da Injecção com as unidades descritas assinaladas.

Legenda: **1.** Microcontrolador; **2.** Sensor de medição de pressão absoluta na admissão; **3.** Interface de I/O (LIN, canal de sincronização e alimentação; **4.** Interface de I/O (reencaminhamento LIN, sensores e actuadores); **5.** Interface de I/O (sensores e actuadores); **6.** Interface de programação do microcontrolador.

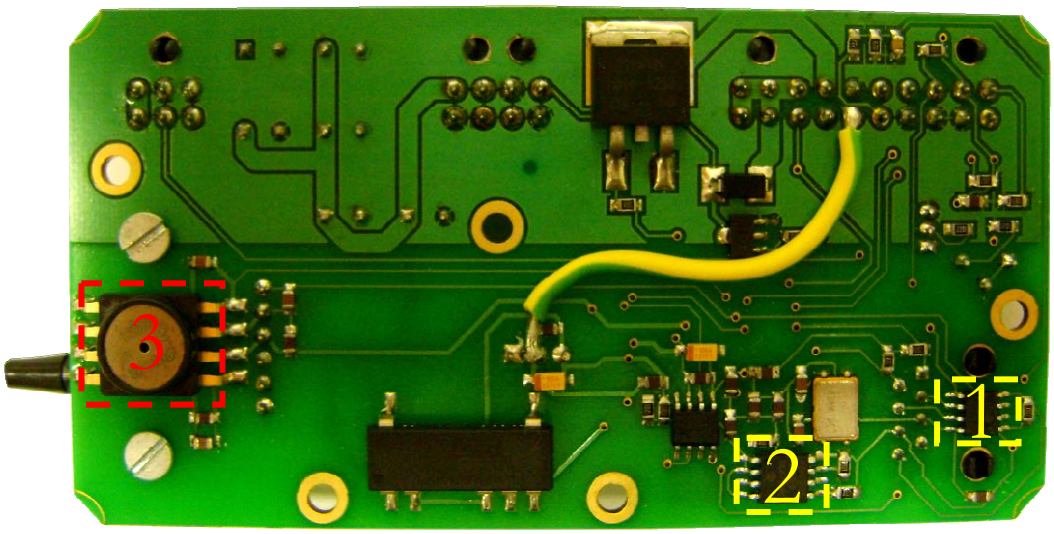


Figura 13 - Aspecto real posterior do Módulo de Controlo da Injecção com as unidades descritas assinaladas.

Legenda: **1.** Microcontrolador auxiliar (canal de sincronização); **2.** Unidade de memória auxiliar EEPROM; **3.** Sensor de medição da pressão atmosférica.

- **Microcontrolador PIC18F6722:** constitui a unidade de processamento principal e a com mais funcionalidades entre todos os módulos (consequência da exigência computacional associada a este módulo);

- **Microcontrolador PIC12F683:** unidade de processamento secundária (de complexidade reduzida), implementada exclusivamente para a produção do sinal de sincronização entre o Módulo de Controlo de Ignição e o Módulo de Controlo de Injecção[8] (*vide Figura 9*);

- **Memória EEPROM:** representa a unidade auxiliar de armazenamento para as tabelas de configuração associadas ao controlo da ignição e injecção do motor;

- **Comunicação LIN:** à semelhança do Módulo de Controlo de Pressão do Combustível, também este se encontra conectado ao barramento LIN e executa uma *Slave Task*. Constitui assim um LIN *Slave* permitindo-o obter e partilhar informação com os restantes módulos;

[8] A informação relativa ao sinal de sincronização é apresentada e discutida posteriormente no ponto *2.3.6*.

- **Controlo do Injector:** o controlo do *timing* da injecção e a quantidade de combustível injectado em cada momento no motor são obtidos através da actuação directa sobre o injector instalado;

- **Controlo do Servo da Borboleta:** actuando directamente sobre o servo instalado na borboleta consegue-se controlar a quantidade de ar que entra para a admissão e consequentemente a aceleração do motor;

- **Controlo do Motor de Arranque:** o accionamento do motor de arranque está atribuído a este módulo, e consequentemente a inicialização do motor de combustão;

- **Medição de Temperaturas:** efectua a medição da temperatura na cabeça e no óleo do motor, e ainda no colector de admissão de ar;

- **Medição de Pressões:** este módulo é igualmente responsável pela medição da pressão atmosférica e sobretudo pela leitura da pressão do ar à entrada da admissão do motor.

Como se pode inferir, as funções delegadas a este módulo de controlo são extensas e essenciais, implicando uma responsabilidade acrescida na sua implementação. Desde o controlo da aceleração, do *timing* e da quantidade de combustível injectado, até à medição de diversos parâmetros de pressão e temperatura, a importância deste módulo é inequívoca.

É novamente devido ao barramento LIN que se torna possível adquirir um estado distribuído de operação dentro do sistema, e deste modo partilhar e controlar os parâmetros apresentados. Resta referir a importância do canal de sincronização adicional, controlado pela unidade de processamento secundária existente no módulo, e que se revelará fundamental para o sincronismo entre a injecção e a ignição.

2.1.4. Módulo de Controlo da Ignição

Subsiste por fim a apresentação e explanação do Módulo de Controlo da Ignição. Sendo a ignição uma componente fulcral e temporalmente bastante exigente (*vide 1.6.1.1*), a tarefa a desempenhar por esta unidade cinge-se à actuação na vela de ignição de forma a cumprir com rigor os requisitos impostos. É possível observar na *Figura 14* o aspecto real do módulo com os principais blocos funcionais destacados e a respectiva legenda, descrevendo-se ainda as suas características:

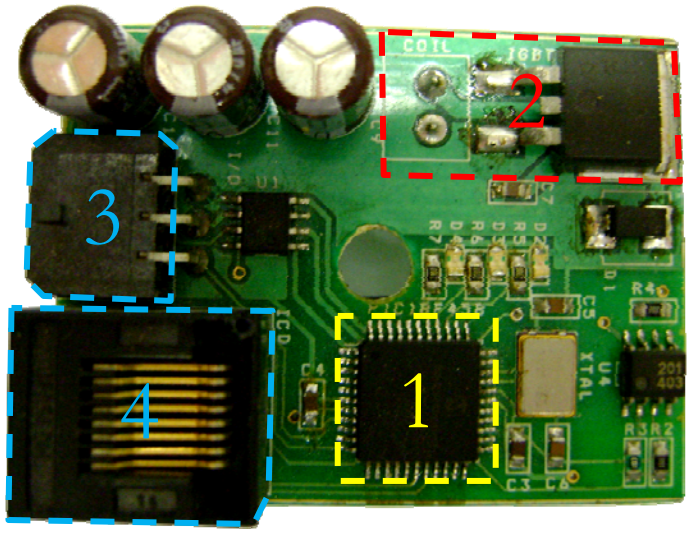


Figura 14 - Aspecto real do Módulo de Controlo da Ignição com as unidades descritas assinaladas.

Legenda: **1.** Microcontrolador; **2.** Interface e actuador de controlo da ignição; **3.** Interface de I/O (LIN, canal de sincronização e alimentação); **4.** Interface de programação do microcontrolador.

- **Microcontrolador PIC18F458:** representa a única unidade de processamento instalada no módulo, análoga à existente no Módulo de Controlo de Pressão de Combustível;
- **Comunicação LIN:** a conexão ao barramento LIN encontra-se implementada neste módulo à semelhança dos anteriores, executando uma *Slave Task*, de forma a possibilitar a partilha de informação e a alteração às suas configurações;
- **Controlo da Vela de Ignição:** actuando directamente sobre vela de ignição, é efectuado o controlo da combustão da mistura ar-combustível no interior do motor.

Tal como foi referido, as funções desta unidade são reduzidas de forma a satisfazer as necessidades temporais associadas. Além da ligação ao barramento LIN que possibilita a conexão aos restantes elementos de controlo da centralina, é a este módulo que se encontra conectado o canal de sincronização dedicado. No ponto seguinte é esclarecida a importância da sua implementação, e explanados os detalhes funcionais da mesma.

2.1.5. Canal de Sincronização Dedicado

Nos pontos anteriores foram apresentadas e detalhadas as características de todos os módulos constituintes do sistema distribuído de controlo electrónico do motor de combustão interna. Foi supracitado igualmente em *1.6.1.1* a importância do controlo do *timing* da ignição e do controlo da injecção no âmbito do consumo de combustível, da emissão de gases de escape e da força produzida pelo motor. Nesse sentido, os módulos com maior relevância são o Módulo de Controlo da Ignição e o Módulo de Controlo da Injecção que efectuam o controlo directo sobre esses parâmetros.

No seguimento desta ilação é necessário conjugar a interdependência existente entre os módulos referidos, e o facto de ambos constituírem unidades de controlo fisicamente dissociadas como parte de um sistema distribuído de controlo electrónico. Deste modo, a sincronização temporal entre estas unidades reveste-se de importância fulcral no desempenho das tarefas associada a cada uma delas, e sobretudo no cumprimento do objectivo global do sistema. Mais concretamente, o conhecimento da posição da cambota (e consequentemente do êmbolo) e da fase do ciclo de combustão não estão directamente acessíveis ao Módulo de Controlo da Ignição. Para tal – e como foi descrito anteriormente em *1.7.2* – é necessário recorrer ao Sensor Angular de Indução e ao Sensor de Pressão de Ar, que se encontram localizados no Módulo de Controlo da Injecção. Considerando este facto torna-se imprescindível a partilha dessa informação entre os dois módulos, surgindo de imediato um problema: como interligar de forma eficaz e segura os dois módulos cumprindo os requisitos temporais exigidos?

Da problemática da implementação de um canal de conexão para o estabelecimento do sincronismo referido, subsistem duas soluções: utilização do barramento LIN já existente ou criação de uma linha dedicada para o efeito. O recurso ao barramento LIN é de imediato limitativo: dependeria da execução da LIN *Schedule* no Módulo de Controlo Central colocando em causa o cumprimento atempado da tarefa; a complexidade de funcionamento do LIN é de sobremaneira desnecessária para a aplicação que se pretende obter; a imposição temporal da tarefa implicaria uma sobre ocupação absurda do barramento.

A partir dos argumentos descritos é necessária a implementação de um canal dedicado de sincronização entre os dois módulos. Desta decisão advêm algumas implicações técnicas: aplicação de um microcontrolador dedicado para a medição da posição do êmbolo e variação da pressão de ar na admissão, e a partir desses dados inferir a fase do ciclo de combustão com precisão; geração de um sinal específico e adaptado ao tipo de informação que se pretende transmitir.

A utilização de um microcontrolador auxiliar justifica-se devido à necessidade de minimização da carga computacional nas unidades principais. Deste modo, consegue-se obter um sinal processado contendo a informação necessária a sinalizar o início de cada novo ciclo e os momentos exactos da injecção e ignição. Obtém-se deste modo, o sinal necessário para estabelecer o sincronismo entre os dois módulos e consequentemente entre as duas tarefas enunciadas.

2.1.5.1. Características Técnicas

São agora apresentadas as características técnicas do canal dedicado de sincronização e o esclarecimento da sua implementação. Inicialmente é necessário entender o funcionamento do Sensor Angular de Indução apresentado em *1.7.2* e o formato do sinal à sua saída. Na *Figura 15* é possível observar uma representação da forma desse sinal.

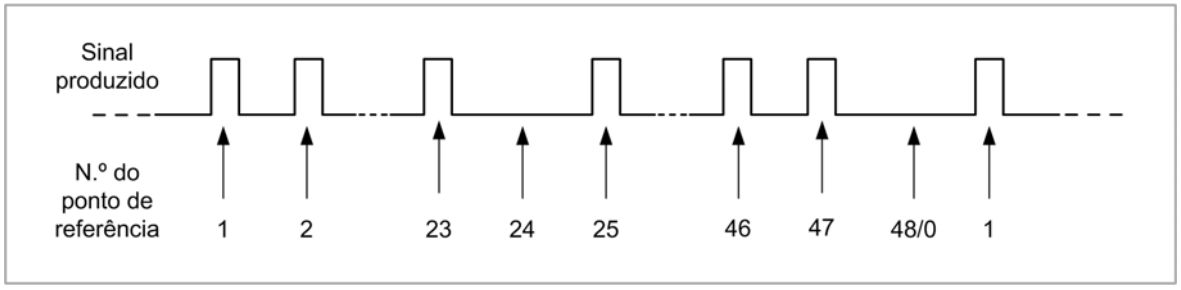


Figura 15 – Representação do sinal gerado pelo Sensor Angular de Indução e a numeração correspondente de cada ponto de referência

É possível observar na representação gráfica a numeração atribuída a cada ponto de referência instalado para a medição da posição do êmbolo. Como já foi referido anteriormente, num motor com funcionamento a quatro tempos são efectuadas duas rotações completas durante a execução de um ciclo completo. Assim, obtêm-se quarenta e oito pontos de referência (vinte e quatro por cada rotação) que podem ser divididos pelas quatro fases do ciclo: 48/0[9] ao 11 – Ignição; 12 ao 23 – Exaustão; 24 ao 35 – Admissão; 36 ao 47 – Compressão. Deste modo, a falta dos pontos de referência correspondentes aos números 24 e 48/0 sinalizam o momento em que o êmbolo atinge o TDC, e portanto, completa um ciclo completo. No entanto, esta informação é insuficiente para inferir acerca da fase do ciclo em

[9] Note-se a justaposição do ponto de referência 48 com o ponto 0, uma vez que o final de um ciclo coincide com o início de um novo.

que se encontra o motor quando o êmbolo atinge um dos TDCs (correspondente ao número 24 ou 48/0); pode estar no início da fase de admissão ou no momento da ignição. Como foi supramencionado, essa informação adicional é obtida a partir do Sensor de Pressão de Ar instalado no colector da admissão, permitindo dessa forma conhecer com exactidão a fase em que se encontra o motor. Sabendo que a detecção de um dos TDCs se traduz no início da fase de admissão e que nessa fase há uma alteração da pressão no interior do colector de admissão (devido à entrada forçada da mistura ar-combustível para a câmara de combustão), através da monitorização da variação dessa pressão aquando da ocorrência de um TDC consegue-se saber indubitavelmente a fase do motor.

O sinal que será expectável alcançar terá de incluir toda a informação necessária ao sincronismo entre os módulos, e por isso, incluir um indicador do início de um novo ciclo. Isto possibilita a produção de um sinal semelhante ao representado na *Figura 15* possuindo uma pequena – mas decisiva – alteração: o ponto de referência número 48/0 (correspondente ao fim/início de um ciclo e depois de ser correctamente detectado) será sinalizado através da criação de um impulso com o dobro da duração temporal relativamente aos restantes impulsos. Por outro lado, o ponto de referência número 24 passa igualmente a ser sinalizado, justificado pela predição do seu acontecimento. Note-se que o início da transmissão do sinal de sincronização final apenas é produzido quando o microcontrolador tem conhecimento exacto da posição do êmbolo e da fase em que se encontra o motor. Na *Figura 16* é possível visualizar uma representação do sinal de sincronização final, a indicação de cada fase e o sinalizador de início de um novo ciclo (ponto de referência número 48/0 com o dobro da duração que os restantes):

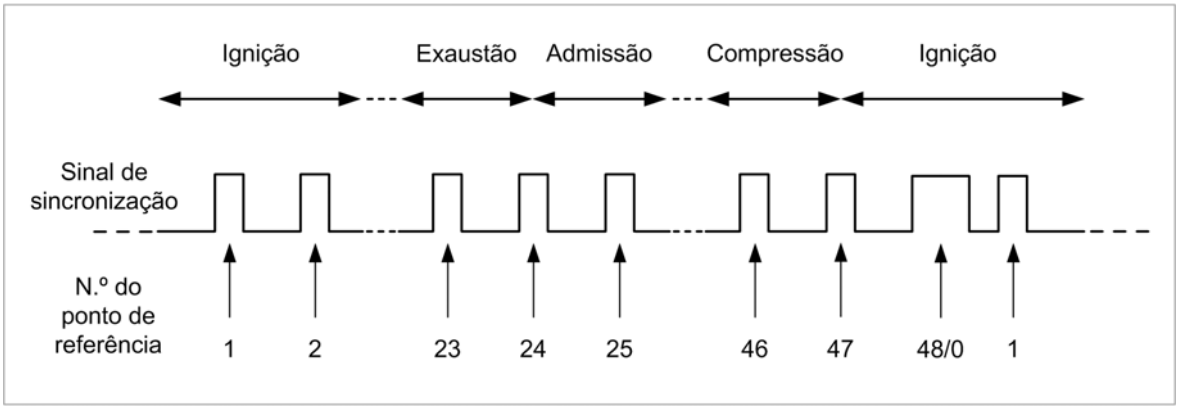


Figura 16 – Representação do sinal de sincronização final, a numeração dos pontos de referência e a indicação correspondente a cada fase de um ciclo.

2.1.5.2. Solução Implementada

Na prática, a detecção e cálculo da fase em que se encontra o motor é realizado pelo microcontrolador secundário instalado no Módulo de Controlo da Injecção. A detecção dos impulsos provenientes do Sensor Angular de Indução é efectuada utilizando a entrada de interrupção existente no microcontrolador PIC12F683. Recorrendo ao uso de um *timer*, é realizado o registo do intervalo de tempo entre cada interrupção, sendo esta uma informação útil na distinção e predição dos pontos de referência número 24 e 48/0. Assim, através de um contador que a cada interrupção incrementa o número do ponto de referência, conjugado com os dois sinalizadores de TDC e o valor da pressão do ar no interior da admissão, o microcontrolador efectua as correcções necessárias para saber inequivocamente qual o ponto de referência correspondente a uma dada interrupção. Com a actualização permanente desta informação, procede-se à sua transmissão para os microcontroladores principais de ambos os módulos.

Do ponto de vista dos Módulos de Controlo da Ignição e Injecção, a recepção do sinal de sincronização como forma de contagem dos pontos de referência apenas é considerada após a detecção do indicador de início de um novo ciclo. A partir desse momento atinge-se a sincronização pretendida entre a ignição e a injecção, procedendo-se à contagem sincronizada dos pontos de referência e consequentemente das fases que constituem um ciclo a quatro tempos.

A duração dos sinais gerados e a detecção dos mesmos são realizadas recorrendo aos *timer modules* existentes nos microcontroladores utilizados. Devido à dinâmica associada na detecção do Sensor Angular, as restrições temporais são elevadas com implicações na configuração dos *timers* referidos. Tome-se como exemplo que para o funcionamento do motor a 12000 RPM (valor dificilmente atingido durante o normal funcionamento do motor), o Sensor Angular de Indução detecta um ponto de referência a cada 208μs. Deste modo, baseando-se nesse valor e na sua ordem de grandeza, decidiu-se atribuir a duração de 50μs aos pontos de referência mais curtos (do 1 ao 47), e 100μs ao ponto de referência indicador de um novo ciclo (48/0). A exactidão na duração dos impulsos é crucial pois dela depende a distinção entre o impulso de sinalização do início de um novo ciclo e os restantes impulsos. Também na sua detecção esse rigor é essencial, nomeadamente para calcular o número de rotações por minuto (RPM) e controlar o momento exacto da ignição e injecção do combustível.

Concluindo, a delegação do cálculo da posição do êmbolo e a determinação da fase de um ciclo são atribuídas a uma unidade de processamento secundária de forma a atenuar a carga computacional dos microcontroladores principais. A implementação do canal de sincronização dedicado representa, por isso, uma característica capital no desenvolvimento de uma solução distribuída para o controlo electrónico de um motor de combustão interna.

Capítulo 3

3. Implementação Ícaro LIN

No projecto Ícaro a implementação do protocolo LIN sofre algumas alterações na LIN *frame* – particularmente na *Frame Header* – resultantes da adaptação deste protocolo no contexto do projecto, designadamente no envio de comandos. Será esclarecido em pormenor o modo de operação da LIN *Master Task* e *Slave Task*, tendo sido concebidos fluxogramas de software com o objectivo de elucidar e alargar a compreensão da solução implementada.

3.1. Adaptações Específicas do Protocolo ao Projecto

Uma vez que o LIN será usado frequentemente na transmissão de mensagens de controlo e sinalização, ou seja, mensagens muito curtas e com pouca informação, optou-se por criar um campo dedicado ao envio deste tipo de comandos. Além de possibilitar a incorporação de instruções na frame *header* (que devido ao seu tamanho e simplicidade não justificam a utilização de *Data Bytes*), também permite o envio de múltiplos *Data Block*, incluindo-se um byte adicional para a identificação de cada bloco. Assim, além de toda a estrutura anteriormente explanada, são adicionados mais dois SCI bytes à *frame header* – o *Command Byte* e o *Index Byte* (sendo este opcional). Adicionalmente, o ID *Byte* sofre algumas alterações no seu formato, diminuindo-se o número de bits de identificação (cinco bits permitindo endereçar 32 destinatários), um bit de sinalização da direcção da mensagem (envio ou solicitação de dados), e dois bits de *Checksum* para detecção de erros.

Na *Figura 17* é possível observar a nova estrutura da LIN *frame*, com as respectivas modificações incluídas. O *Command Byte* permite, portanto, o envio de instruções específicas (comandos de controlo) sem a necessidade da utilização dos *Data Bytes* no envio dessa instrução (utilizam-se os *Data Bytes* para envio da resposta ou simplesmente para *Acknowledge*), possibilitando o encurtamento das mensagens e consequente libertação do *bus*. O campo *Mbb* (*Multi Block Byte*) incluído no *Command Byte* é usado para indicar o envio de apenas uma, ou múltiplas *Frame Response* consecutivas pertencentes ao mesmo conjunto de informação. No caso da transmissão de várias *Frame Response* é necessário incluir no *Frame Header* o *Index Byte*, cuja finalidade é assinalar qual o índex correspondente à *Frame Response* que será enviada. À semelhança do ID *Byte*, também o *Command Byte* inclui dois bits de *Checksum* para a detecção de falhas.

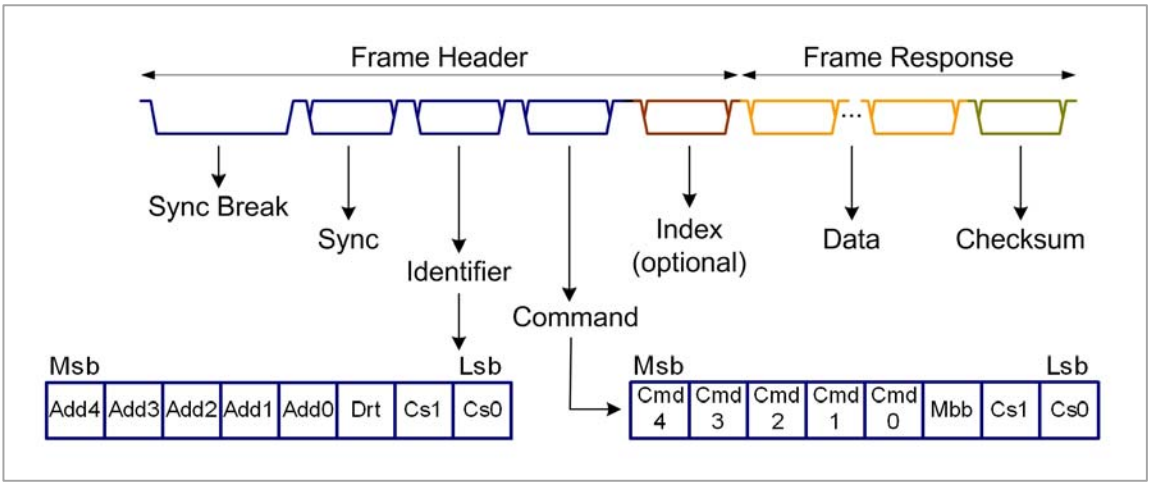


Figura 17 – Representação da LIN *frame* com as alterações descritas.

Estas modificações permitem ajustar o modo de funcionamento do protocolo e o formato das mensagens ao tipo de comunicação pretendida entre os vários ECUs que controlarão o funcionamento de todo o sistema. Cria-se deste modo um protocolo de comunicação à medida do projecto, a partir de um conjunto de especificações pré-existentes, possibilitando a optimização do bus de comunicação.

Como foi supracitado em *1.5.1*, o protocolo LIN apenas necessita das SCI presentes em todos os microcontroladores usados no projecto que lidam com o protocolo. No entanto, é inevitável a utilização de um componente adicional que adapte o contexto de operação da SCI do microcontrolador às especificações do protocolo. Sendo o LIN um protocolo com comunicação *half-duplex* (utiliza apenas uma linha de transmissão), é necessária a utilização de um *transceiver* que possibilite a conexão do microcontrolador ao barramento (através dos pinos de transmissão e recepção da SCI). Utilizou-se para isso o MCP201, que consiste num "*LIN Transceiver with Voltage Regulator*", cujas funções incluem a passagem da voltagem de operação da SCI do microcontrolador (5V) para uma voltagem de funcionamento superior (neste caso a voltagem de alimentação dos módulos – 12V), a implementação da resistência de *pull-up*, e ainda outras funcionalidades (protecções contra curto-circuitos na linha de comunicação, protecção contra sobreaquecimento, etc.). Além disso, incorpora um regulador de voltagem especificamente projectado para operar no meio automóvel, suportando trocas de conexões da bateria e picos de tensão. O MCP201 actua, portanto, como um agente intermédio entre o ambiente de operação do microcontrolador (e da sua SCI) e o meio do barramento de comunicação LIN.

Passando à análise do protocolo LIN na perspectiva do microcontrolador, foi necessário configurar a SCI de cada um de acordo com as especificações indicadas nos respectivos *datasheets* [20]. A taxa de transmissão de dados definida foi de 19.2Kbps em modo assíncrono, permitindo utilizar o transmissor e o receptor da SCI de forma funcionalmente

independente, operando em modo *full-duplex* (embora no panorama do barramento LIN continue a ser *half-duplex*), permitindo dessa forma executar no *Master Node* a *Slave Task* e a *Master Task* simultaneamente (*vide 1.5.1.1*).

Segue-se a exposição em detalhe do software implementado e a descrição da LIN *Master Task* e *Slave Task*.

3.2. Solução Implementada

Expostas as alterações efectuadas ao protocolo LIN e as especificações de hardware necessárias ao seu funcionamento, procede-se à descrição detalhada do software desenvolvido. O atendimento às interrupções referidas é realizado através da execução de uma ISR – *Interrupt Service Routine* (Rotina de Serviço à Interrupção), e toda a informação enviada ou recebida é temporariamente armazenada em dois *buffers* circulares distintos para a recepção e transmissão de mensagens (*Input Buffer* e *Output Buffer* respectivamente). O manuseamento destes *buffers* é realizado por rotinas específicas, do mesmo modo que foram implementadas funções para a gestão e acondicionamento da informação enviada ou recebida.

Como foi referido anteriormente, a recepção e transmissão de mensagens (enchimento dos *buffers*) é efectuado utilizando as interrupções associadas existentes nos microcontroladores. No entanto, a LIN *Schedule* e o respectivo agendamento das comunicações são executados dentro da função principal *main*, assim como a verificação de novas mensagens nos *buffers* de entrada/saída e o respectivo processamento. É necessário, por isso, diferenciar a execução da *Master Task* da *Slave Task* e compreender o mecanismo de operação de ambas. Consistindo em tarefas operacionalmente distintas, originam portanto, soluções de software igualmente diferentes. A solução desenvolvida para a *Slave Task* é baseada na execução periódica de uma máquina de estados, enquanto que a *Master Tak* é executada segundo o agendamento programado da LIN *Schedule* ou por ocorrência de eventos externos. É necessário referir que a implementação da *Slave Task* no *Master Node* é diferente das existentes nos *Slave Nodes*. As especificações sobre o funcionamento detalhado das *Tasks* são apresentadas de seguida.

3.2.1. LIN *Master Task*

A execução da LIN *Master Task* é realizada de acordo com o agendamento da troca de mensagens entre os módulos (*e.g.* actualização periódica do valor de RPM), ou pelo

acontecimento não programado de um evento (*e.g.* paragem do motor). Sendo responsabilidade do *Master* o despoletar das comunicações, é necessário o preenchimento da estrutura do *Frame Header* para posterior transmissão. Para esse fim foi criada a função *FillLINFrameStruct()* cujos parâmetros de entrada são todos os elementos configuráveis e essenciais à constituição da *Frame Header*, excepto o *Sync* Byte que é fixo. Define-se portanto, o ID do módulo destinatário, o comando pretendido, a direcção da mensagem (pedido ou envio de informação para um *slave*), o *Multi Block Bit*, e o *Index* correspondente caso se tratem de *Multi Block Messages*. São ainda calculados os bits de *Checksum* correspondentes para o ID *Byte* e o *Command* Byte. A função descrita tem necessariamente de ser invocada e introduzidos parâmetros válidos, antes de carregar o *Frame Header* para o *Output Buffer* e iniciar-se a transmissão. Esta última tarefa é realizada invocando uma função específica que escreve toda a informação guardada na estrutura do *Frame Header* para o *Output Buffer*, e consequentemente activando a interrupção respectiva. Uma terceira função é invocada na ISR efectuando o *dispatch* da informação para o barramento.

É importante notar que a transmissão do *Sync Break* não é efectuada por software, e portanto não existe um campo na estrutura mencionada para esse efeito. Esta operação é realizada por hardware e permitida devido ao uso do microcontrolador PIC18F4585, cujas características incluem uma SCI com suporte para o protocolo LIN 1.3, nomeadamente na transmissão do *Sync Break* (consiste no envio de um *Start bit*, seguido de doze bits a '0' e um *Stop bit*). Basta para isso activar o módulo de transmissão da SCI e o bit de envio do *Sync Break Character* (SENDB), que é de seguida automaticamente reinicializado por hardware possibilitando o pré-carregamento do módulo de transmissão com o byte seguinte. Usufrui-se deste modo, de uma funcionalidade intrínseca ao microcontrolador utilizado, encontrando-se preparado para suportar a implementação do protocolo LIN.

Finaliza-se assim, o processo de envio de uma *Frame Header* e a execução da LIN *Master Task*, iniciando-se a transmissão de uma mensagem no barramento LIN. Seguidamente, de forma a concluir todo o processo de comunicação, é abordada a *Slave Task* e explicado o princípio de funcionamento da solução desenvolvida.

3.2.2. LIN *Slave Task*

A implementação da LIN *Slave Task* implicou o desenvolvimento de duas soluções de software distintas, como resultado da adaptação ao funcionamento dos dois elementos básicos do protocolo LIN – o *Master* e o *Slave*. Como foi mencionado anteriormente, o *Master Node* possui duas tarefas, a *Master e a Slave Task*, enquanto que no *Slave Node* apenas é executada a *Slave Task* (*vide Figura 4*). Assim, o *Master* opera em modo *full-duplex* na transmissão ao enviar e receber em simultâneo a *Frame Header*, comportando-se como um *Slave*. Existe uma dissociação entre as duas *Tasks* contidas no *Master*, sendo que para a transmissão de

informação este necessita de enviar uma *Frame Header* a si próprio. Considerando este facto, depreende-se que é o *Master* a despoletar a transmissão de todas as *Frame Responses*, incluindo as das suas próprias mensagens. Não obstante, o facto das duas *Tasks* serem executadas num mesmo nó, exigiu abordar o trabalho desenvolvido de um modo diferente. Deste modo, foram criadas duas funções distintas – *ProcessLINMaster()* e *ProcessLINSlave()* – para a implementação da *Slave Task* no *Master Node* e no *Slave Node*, respectivamente.

Resta ainda referir o modo como é efectuada a recepção e validação da *Sync Break* nos microcontroladores. Uma vez que o formato da *Sync Break* não permite a sua detecção como uma mensagem de comunicação serie válida (*Start bit*, oito bits de informação, *Stop bit*), foi necessário recorrer a uma funcionalidade incorporada na SCI dos microcontroladores utilizados. Usufruiu-se portanto, do bit FERR (*Framing Error Bit*) indicativo de erro na constituição da *frame* recebida, possibilitando desta forma a identificação da *Sync Break*.

3.2.2.1. Implementação *Master*

A *Figura 18* apresenta o fluxograma representativo do modo de operação da função *ProcessLINMaster()*, consistindo numa máquina de estados. Deste modo, a primeira condição a ser verificada dentro da função é a existência de mensagens no *Input Buffer* (invocando a função *CheckInBuffer()*).O prosseguimento da execução da máquina de estados depende sempre da existência de informação no *Input Buffer*, implicando que essa condição seja verificada sempre que haja uma alteração de estado. A sua execução é terminada se não existir informação para ser processada, caso contrário é salvaguardado (em *rxChar*) e libertado o primeiro byte contido no *buffer*. De seguida, é testada a ocorrência de uma *Sync Break* permitindo inferir se a informação a ser processada pertence ao início de uma nova mensagem (exigindo o recomeço da máquina de estados); ou se é a continuação dos dados da mensagem anterior (continuando a execução da máquina de estados consultando o estado anteriormente salvaguardado).

A máquina de estados é efectivamente implementada através de uma estrutura de controlo *switch* testando a variável estática *state*. Em todos os estados à excepção do "4", a detecção de alguma incongruência implica reiniciar a máquina de estados e apagar toda a informação contida no *Input Buffer* (função *ClearInputBuffer()*).

Procedendo-se à análise do fluxograma representado na *Figura 18*, observa-se que no estado inicial "0" é verificada a recepção do *Sync Byte* e actualizado o estado para "1" em caso afirmativo. De seguida, no estado "1", é verificada a validade do campo de ID comparando os dois bits de *Checksum* incluídos com o valor calculado, e novamente actualizado o estado. No estado seguinte é testada a validade do comando através de um processo análogo ao anterior, e reconhecida a sua atribuição na lista de comandos. A próxima condição de teste é realizada ao bit *Mbb*, identificando a mensagem como *Single* ou *Multi Block Message*. No caso de ser *Multi*

Block prossegue-se para o estado seguinte sem alterações, caso contrário verifica-se em que sentido será enviada a *Frame Response*. Se os dados forem enviados no sentido *Master-Slave* é invocada a função *ProcessOutMessage()* que é responsável pela realização da tarefa associada ao comando recebido. Se o sentido de transmissão for inverso o estado é alterado para "4".

O estado "3" é executado na condição de se tratar de uma *Multi Block Message* e portanto, o próximo byte a ser processado é o *Index Byte*. O valor máximo de blocos possíveis de ser enviados com tamanho igual a oito bytes é de vinte (pois o tamanho do *buffer* de dados é igual a 160 bytes), sendo necessário verificar se o valor recebido não excede esse limite. Resta testar o sentido de envio da *Frame Response*, invocando a função *ProcessOutMessage()* e reiniciando a máquina de estados no caso de ser o *Master* a transmitir dados. Se o sentido for oposto avança-se para o estado "4" sem alterações adicionais.

É no quarto estado que se efectua o armazenamento da informação transmitida na *Frame Response*, e portanto, de todos os bytes enviadas nesse bloco independentemente de se tratar de uma *Single* ou *Multi Block Message*. A máquina de estados é mantida nesse estado até o byte final ser guardado na estrutura de dados, transitando por fim para o quinto e último estado.

No último estado é calculado o *Checksum* final e comparado com o valor recebido no final da *Frame Response*. Se a correspondência falhar, o conteúdo do *Input Buffer* é apagado. Se por outro lado, a comparação for bem sucedida, é invocada a função *ProcessInMessage()* e executadas as tarefas associadas ao comando recebido. Em ambas as situações a máquina de estados é colocada no estado inicial.

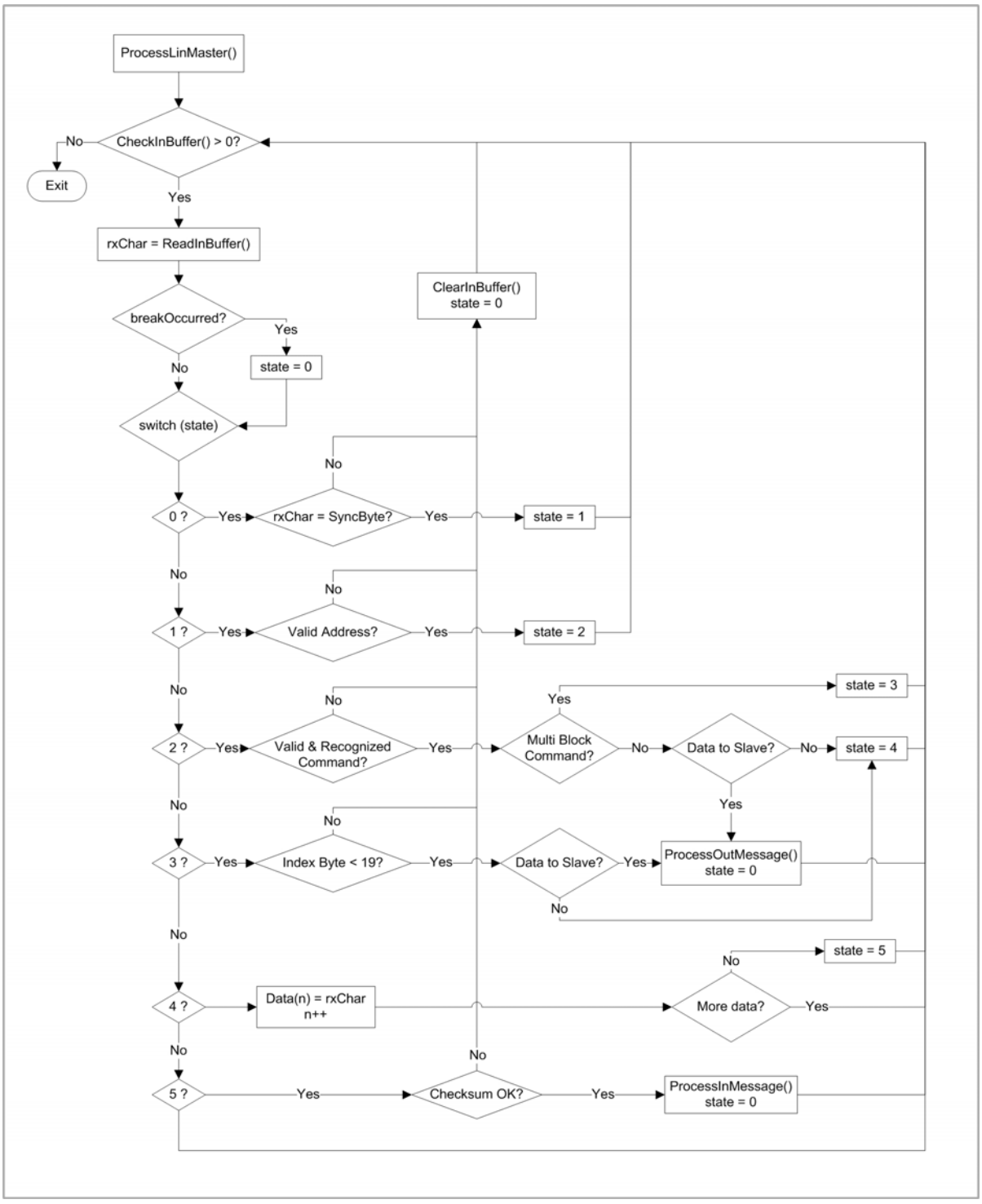


Figura 18 – Fluxograma da função *ProcessLinMater()* executada na LIN *Master Task*.

3.2.2.2. Implementação *Slave*

Tal como foi referido anteriormente, a implementação da *Slave Task* nos módulos *Slaves* requer a consideração de alguns aspectos operacionais distintos relativamente à aplicação no módulo *Master*. A diferença principal reside no facto dos *Slave Nodes* não terem conhecimento dos campos da estrutura correspondente à *Frame Header*. No *Master Node* esta informação é implícita através da execução da *Master Task*, responsável pelo preenchimento da estrutura. Assim, nos *Slaves* esta informação é recebida através do barramento LIN e guardada na estrutura correspondente.

À semelhança da função *ProcessLinMaster()* implementada no *Master*, a função *ProcessLinSlave()* partilha diversas características nomeadamente a operação como máquina de estados. De facto, observando a *Figura 19* onde se encontra representado o fluxograma da função *ProcessLinSlave()*, constatam-se várias similaridades com o da *Figura 18*. Verifica-se que o modo de operação é exactamente o mesmo até ao estado "2". Caso se trate do envio de uma *Multi Block Message* o estado é alterado para um novo sexto estado, inexistente na implementação *Master*. Nesse estado é guardado o valor do *Index* correspondente ao bloco enviado e executa-se a função *ProcessOutMessage()* reiniciando a máquina de estados para a recepção do bloco seguinte.

A transição para o estado "3" é efectuada na condição de recepção de uma *Frame Response*. No caso de ser uma *Single Block Message*, é guardado o primeiro byte de informação e os restantes no estado "4". Terminado o armazenamento de todos os bytes, avança-se para o estado seguinte onde é calculado e verificado o byte de *Checksum*. Como se trata de uma mensagem constituída por um único bloco de informação, é invocada a função *ProcessInMessage()* e reiniciada a máquina de estados. No caso de uma *Multi Block Message* é guardado o *Index* do bloco que está a ser transmitido, e armazenados os bytes correspondentes no estado "4". No estado "5" processa-se o byte de *Checksum* e actualiza-se e estado para "0" (a próxima mensagem contém o bloco de informação seguinte). A função *PorcessInMessage()* apenas é invocada quando for recebido o último bloco de uma *Multi Block Message*.

Verificam-se portanto, algumas diferenças estruturais nas duas funções que implementam as *Slave Tasks* nos dois tipos de nós. Resta abordar sucintamente a importância de duas funções que foram referidas durante as explanações anteriores – *ProcessInMessage()* e *ProcessOutMessage()*. Ambas estão relacionadas com execução da tarefa correspondente ao comando recebido. A primeira é responsável pelo processamento da *Frame Response* recebida, quer seja no *Master* ou num *Slave*. A segunda função executa uma determinada acção com base nos parâmetros da *Frame Header*, envolvendo sempre a transmissão de uma *Frame Response* com a informação correspondente.

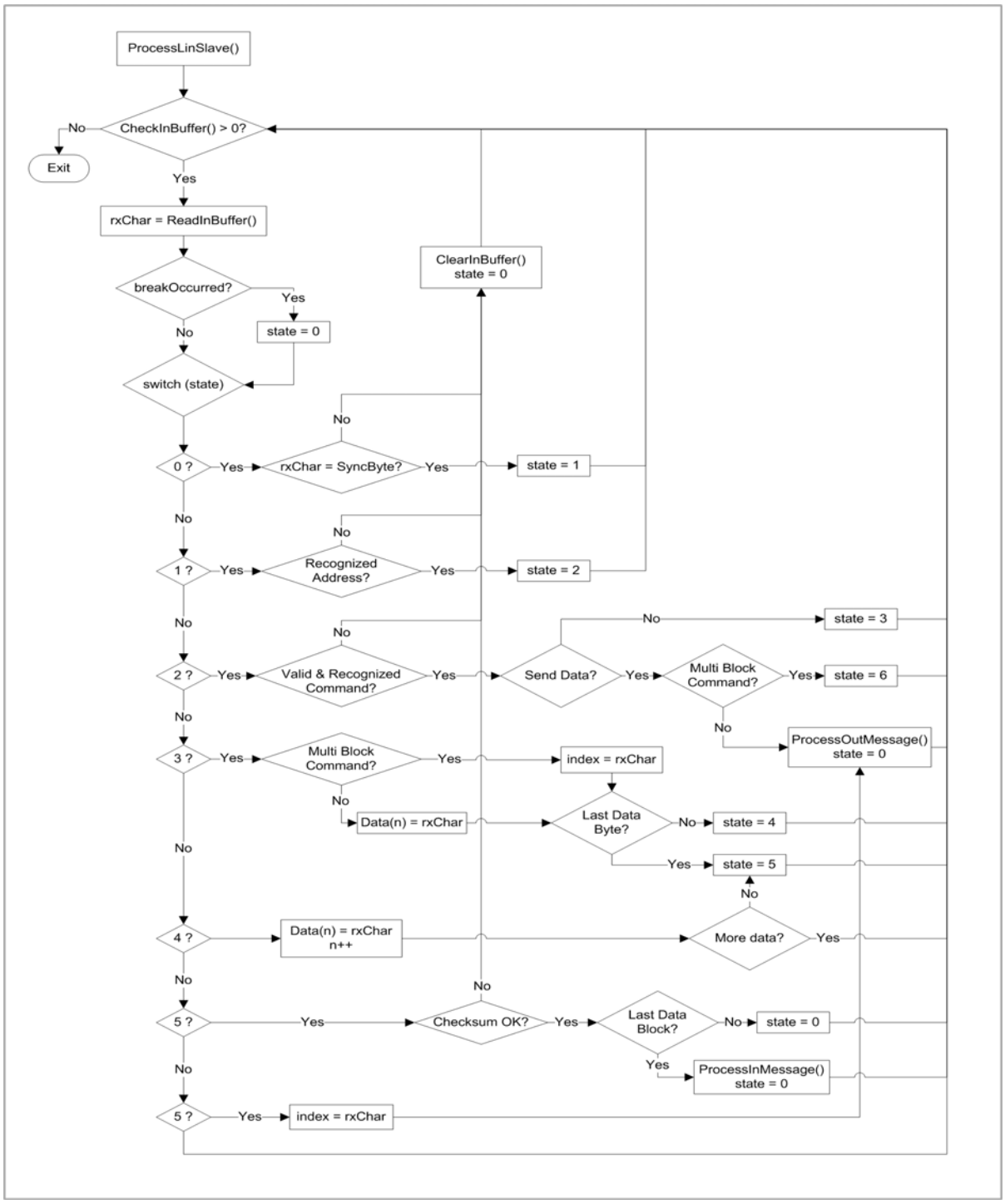


Figura 19 – Fluxograma da função *ProcessLinSlave()* executada na LIN *Slave Task*.

Conclui-se, deste modo, o capítulo correspondente à implementação do protocolo LIN no projecto Ícaro. Foram apresentadas as alterações efectuadas com o objectivo de ajustar este protocolo às exigências agregadas ao controlo electrónico de motores de combustão de interna.

Capítulo 4

4. Interface de Monitorização e Diagnóstico

O controlo electrónico de motores de combustão interna encontra-se necessariamente associado à monitorização de diversos sensores e variáveis, conjuntamente com o accionamento de múltiplos actuadores e tarefas. Estas características justificaram o desenvolvimento de uma interface externa de comunicação, monitorização e diagnóstico do motor – IMD (Interface de Monitorização e Diagnóstico). A necessidade de acompanhamento em tempo real dessas variáveis e tarefas, assim como a reconfiguração de parâmetros do motor, constituem exemplos adicionais que fundamentam esse desenvolvimento. Além disso, a utilização de valores tabelados para o controlo da injecção e ignição requer a existência de uma interface entre a ECU e a unidade computacional que possui essa informação (*e.g.* computador).

4.1. Comunicação USB

Como foi mencionado anteriormente, a interface desenvolvida representa a fronteira de interligação entre a ECU e o computador. Foi necessário, portanto, a implementação de um meio de comunicação compatível com ambos os sistemas. Deste modo, optou-se pelo barramento USB pois representa a interface mais utilizada nos computadores actuais para a conexão de dispositivos periféricos. Apesar da complexidade associada à implementação deste barramento, essa tarefa foi bastante simplificada recorrendo-se ao uso de um módulo conversor de comunicação USB para comunicação paralela. Desta forma obtém-se uma abstracção da comunicação USB em si, focalizando-se o desenvolvimento na comunicação paralelo que possui características funcionais mais simplificadas e acessíveis.

De seguida é brevemente apresentado o módulo conversor utilizado, as suas características e detalhes funcionais. Mais adiante são explicadas as especificidades técnicas do protocolo de comunicação implementado e ainda a solução de software desenvolvida, do ponto de vista da ECU e do computador.

4.1.1. Módulo Conversor USB-Paralelo

O módulo conversor USB-Paralelo utilizado é o UM245R da FTDI (*Future Technology Devices International*). Consiste num módulo de desenvolvimento que incorpora o *chip* FT245RL (circuito integrado conversor USB/USART), um circuito de relógio próprio, e todo o hardware necessário à conexão USB. Na *Figura 20* (obtida em [21]) é possível visualizar o aspecto real do módulo com o respectivo conector USB tipo B, o *chip* FT245RL, dois *jumpers* e a *socket* de encaixe na placa de circuito impresso.

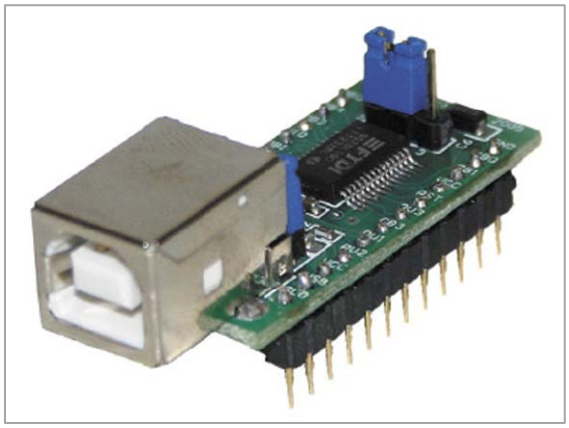


Figura 20 – Aspecto real do módulo conversor USB-Paralelo UM245R.

O módulo conversor suporta dois modos de operação do ponto de vista do computador, isto é, existem duas configurações disponíveis para instalação dos *device drivers* correspondentes: funcionamento como uma porta USB efectiva ou como uma porta de comunicação série virtual (porta COM). Apesar de ser possível obter velocidades de transferência superiores com a primeira opção (1 MByte *vs.* 300 KByte), a segunda solução apresenta uma facilidade de implementação largamente maior. A configuração e o manuseamento de uma porta de comunicação série – à semelhança do que acontece com a SCI dos microcontroladores – foram factores determinantes para a escolha dessa configuração. Além disso, a velocidade de comunicação mais reduzida associada a esta escolha não afecta de sobremaneira o desempenho do sistema a implementar.

Do ponto de vista da ECU – mais especificamente do microcontrolador – a leitura e escrita de dados no módulo é efectuada através do manuseamento dos pinos de dados (DB[7..0]) em conjunto com os pinos de *status* (TXE# e RXF#) e controlo (WR# e RD#).

Recorrendo à representação gráfica do módulo exposta na *Figura 21* (obtida em [21]), é possível observar a configuração de todos os pinos.

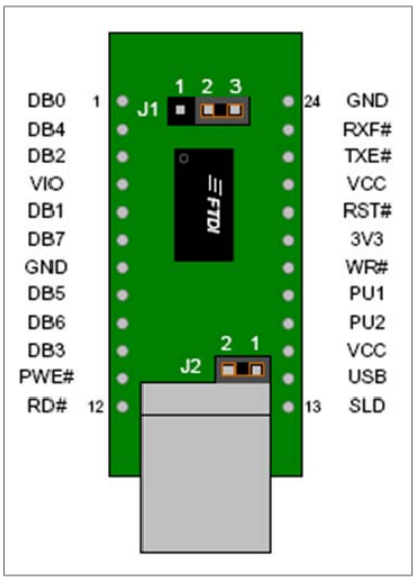


Figura 21 – Representação gráfica da configuração do módulo UM245R.

A leitura e a escrita de dados na estrutura FIFO incluída no *chip* são conseguidas efectuando a conexão dos pinos de dados a um porto completo de I/O (Input/Output) do microcontrolador. Deste modo, facilita-se a implementação do software para a execução destas operações, uma vez que o acesso para leitura/escrita de um porto de I/O é imediato. Além disso, é fundamental o acesso aos pinos de controlo e *status* para a correcta recepção e transmissão dos dados. Os pinos de controlo (WR# e RD#) permitem indicar ao módulo a intenção de colocar ou obter um byte de informação na FIFO de transmissão. Adicionalmente, a disponibilidade de alteração dos dados na FIFO é sinalizada pelos pinos de *status* (TXE# e RXF#), pressupondo a verificação prévia dos mesmos na realização de qualquer operação.

Resta ainda abordar as configurações disponíveis para os *jumpers* incluídos, cuja função é permitir escolher o modo de alimentação do módulo – através da porta USB ou alimentação própria. Neste caso foi utilizada a configuração com alimentação própria obtida a partir do Módulo de Controlo Central.

4.1.2. Características Técnicas do Protocolo Implementado

À semelhança das adaptações efectuadas ao protocolo LIN, também a comunicação implementada entre a ECU e a Interface de Monitorização e Diagnóstico foi desenvolvida especificamente para o tipo de mensagens utilizadas. Não obstante, existem grandes diferenças na estrutura da *frame* das mensagens e no funcionamento da comunicação.

Relativamente à *frame* das mensagens, é possível visualizar na *Figura 22* uma representação do seu formato e uma descrição dos campos constituintes. Cada mensagem é constituída pela *Frame Header* e pela *Frame Response*, tal como acontece com o protocolo LIN. Por sua vez, a *Frame Header* é composto pelo *Header 0* consistindo num byte com todos os bits a zero, servindo como indicador de nova mensagem.

O *Header 1* é formado pelo bit "Snd" que indica se uma determinada mensagem possui *Data Bytes* na *Frame Response*. A existência deste bit justifica-se pelo facto de simplificar o desenvolvimento do software para a comunicação, permitindo saber *a priori* se a mensagem tem bytes de informação associados. Os restantes bits do *Header 1* são utilizados para conhecer a extensão restante da mensagem, e portanto, contribuir para a detecção de falhas.

O campo *Module & Command ID* permite identificar o destinatário ou o remetente (dependendo da comunicação efectuar-se no sentido da IMD para a ECU, ou no sentido inverso), e ainda o comando a ser transmitido. Foram atribuídos três bits para endereçamento dos quatro módulos existentes (permitindo até um máximo de oito), e cinco bits para a distinção dos variados comandos (possibilitando a definição de, no máximo, trinta e dois comandos para cada módulo).

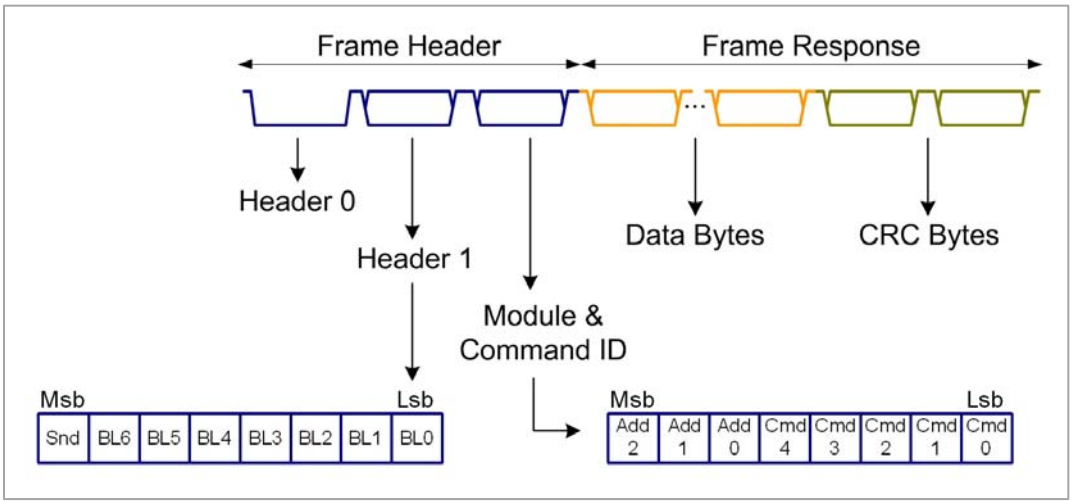


Figura 22 – Representação da *frame* das mensagens do protocolo implementado.

Relativamente à *Frame Response*, esta é formada pelos *Data Bytes*, que variam de acordo com a mensagem enviada e o comando associado, e por dois bytes de CRC (*Cyclic Redundancy Check*) para verificação da integridade dos dados enviados e detecção de falhas. Foi implementado o CRC-16-CCITT (CRC de dezasseis bits criado pelo *Comité Consultatif International Téléphonique et Télégraphique*, actual ITU – *International Telecommunication Union*).

Apesar das semelhanças entre o formato da *frame* das mensagens do LIN e o protocolo apresentado, as características de ambos diferem em variados aspectos. Nomeadamente ao nível da topologia usada, tratando-se esta de um tipo de comunicação ponto-a-ponto, não existe diferenciação entre os dois intervenientes, sendo desnecessárias as definições de *Master* e *Slave*. Ambos podem enviar a *Frame Header* e dar início a uma nova transmissão, possibilitando a troca de informação *on demand* nos dois sentidos.

4.1.3. Solução Implementada

Expostas as características técnicas do protocolo projectado, segue-se a apresentação detalhada da solução de software implementada. É essencial esclarecer que a solução desenvolvida para a comunicação entre a IMD e a ECU, embora seja executada em meios diferentes, é funcionalmente a mesma. As diferenças entre a implementação da comunicação na IMD ou na ECU, resumem-se apenas às adaptações das linguagens utilizadas em cada uma. A diferenciação operacional e comportamental é inexistente, interessando apenas adaptar a solução a cada plataforma e linguagem de programação utilizada. Procede-se então, à explicação das adaptações necessárias para o funcionamento do software nos dois sistemas.

Para a implementação da IMD foi utilizada a linguagem de programação *Visual Basic* juntamente com a *Microsoft .NET Framework* (VB.NET), representando portanto, um desafio na aprendizagem de uma nova linguagem de programação. Os microcontroladores dos módulos de controlo da ECU foram programados em linguagem C. Não obstante, o modo como é armazenada e processada a informação é o mesmo do ponto de vista funcional.

Devido à baixa exigência temporal associada à transmissão de mensagens neste protocolo de comunicação, a implementação da recepção das mensagens na IMD e na ECU, não é efectuada por interrupções (opostamente ao que sucede para o LIN). Optou-se por realizar o atendimento à recepção de mensagens recorrendo à configuração de *timers* que verificam com um período de 50ms o estado de ocupação do buffer de entrada, e posteriormente efectuam o processamento da informação. É indispensável referir que a concretização desta operação deve-se exclusivamente à existência de um *buffer* com estrutura FIFO no módulo conversor USB-Paralelo. O armazenamento temporário da informação no *buffer* intermédio não obriga à utilização de interrupções para a leitura imediata dos dados,

libertando o processamento computacional disponível para tarefas temporalmente mais rigorosas.

Foi explicado anteriormente que a recepção de informação na ECU é realizada através de um porto de I/O do microcontrolador, juntamente com sinais de controlo e *status*. Deste modo, as operações de escrita e leitura no *buffer* FIFO incluído no módulo conversor são executadas actuando directamente nos pinos de dados e controlo, com a monitorização prévia dos sinais de *status*. Por outro lado, na IMD recorreu-se ao uso de uma porta série virtual (COM) cujo manuseamento é efectuado utilizando funções do sistema disponíveis para o efeito. Usufrui-se, deste modo, de uma abstracção na leitura e escrita da informação facilitando consideravelmente a implementação dessas tarefas.

Como foi referido anteriormente, o início da transmissão de uma mensagem pode ser efectuado nos dois sentidos da conexão. A função *FillUSBFrameStruct()* foi criada para realizar o preenchimento da estrutura que contém os campos da *Frame Header*. De seguida, a função *USBInsertHeader()* é responsável pela colocação desses campos no *buffer* de saída, efectuando também o preenchimento do bit *Snd* sinalizando o destinatário sobre a existência de dados associados à mensagem enviada.

A função de processamento da recepção de mensagens foi implementada através de uma máquina de estados, à semelhança da solução desenvolvida para o protocolo LIN. Na *Figura 23* observa-se o fluxograma da função *USBProcessMessage()*, responsável pelo acondicionamento dos dados recebidos.

Procedendo-se à explicação do fluxograma apresentado, observa-se inicialmente o teste de uma condição antes da execução da máquina de estados (a *flag initTX*). Tratando-se de um protocolo com comunicação bidireccional, esta verificação é necessária para saber quem iniciou a transmissão da mensagem – a IMD ou o módulo. A alteração do estado para "3" é efectuada se já tiver sido transmitida a *Frame Header* (sinalizada através da *flag initTX*) e se esteja a aguardar a recepção da *Frame Response*, procedendo-se ao armazenamento dos dados e à execução do CRC da mensagem.

A execução da máquina de estado é iniciada e continuada com a verificação de informação no *Input Buffer*. De seguida, no primeiro estado confere-se a recepção do *Header 0* avançando para o estado seguinte em caso de sucesso. Caso contrário, verifica-se a ocorrência de um erro na transmissão, provocando o apagamento da informação no *Input Buffer* e o reiniciar da máquina de estados.

O estado seguinte corresponde à recepção do *Header 1* que contém o bit *Snd* sinalizando a existência de dados associados à mensagem recebida, e os bits identificadores da extensão da mensagem. Neste estado ainda não é possível verificar se esta informação corresponde ao tamanho da mensagem associado ao comando transmitido, um vez que só no próximo estado se conhece qual o comando. Desta forma, apenas se efectua a salvaguarda da informação transmitida no *Header 1*.

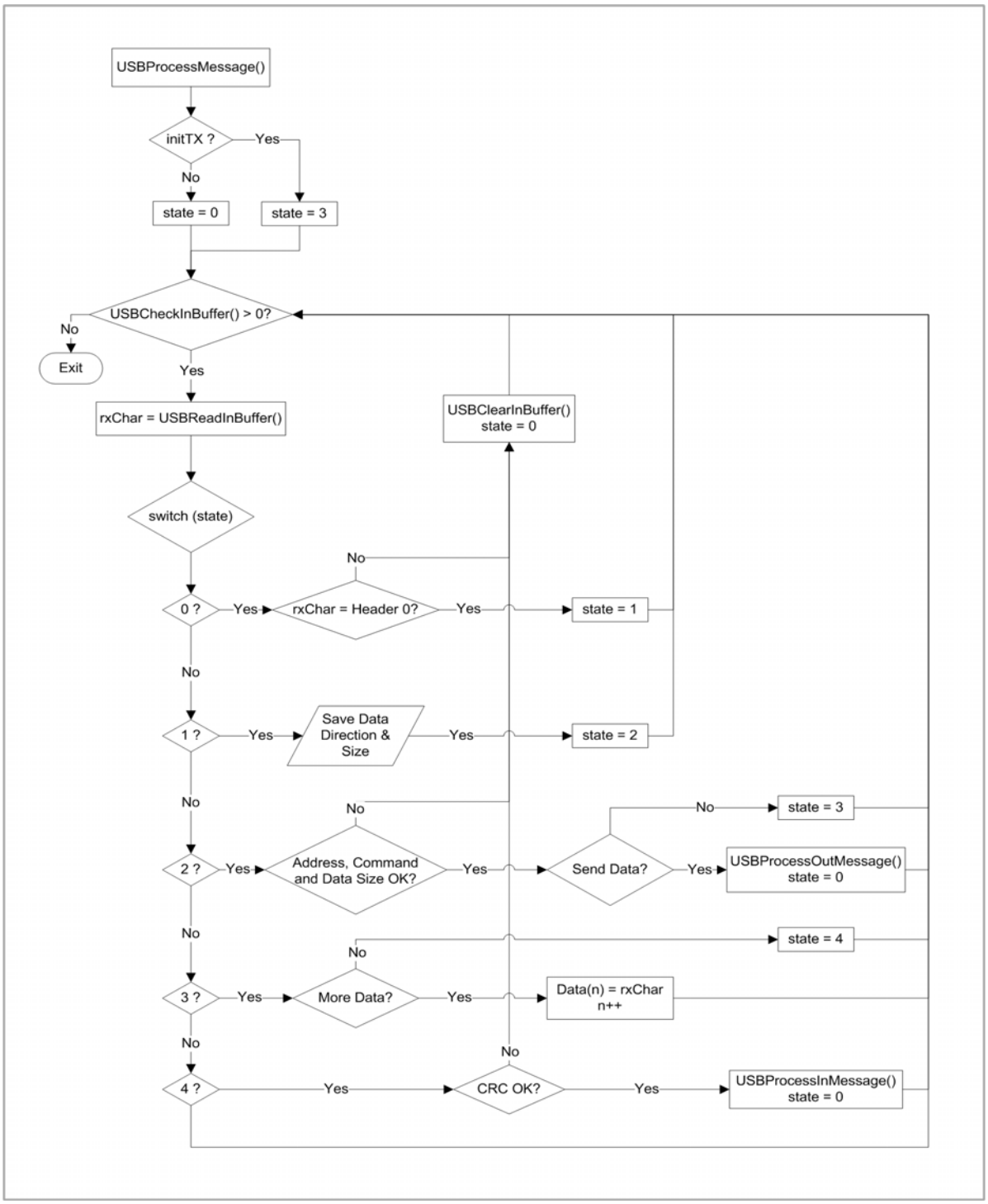


Figura 23 – Fluxograma da função *USBProcessMessage()* executada na IMD e na ECU.

No estado "2" é recebida a informação contendo o destinatário da mensagem e a identificação do comando. Assim, é verificada a validade de ambos e confirmado o tamanho da mensagem transmitido pelo *Header 1*. Em caso de erro em algum dos campos, é executado

o mesmo procedimento que no estado "0". De seguida, se existirem dados a ser enviados, é executada a função *USBProcessOutMessage()* efectuando a tarefa associada à mensagem recebida. Se não existirem dados para ser transmitidos, o estado é actualizado para "3".

O armazenamento dos dados de informação transmitidos é efectuado no estado "3". A execução da máquina de estados é mantida nesse estado até terminar o armazenamento de todos os *Data Bytes*, momento em que transita para o estado seguinte.

No último estado é confirmada a validade e correcção da mensagem recebida calculando os bytes de CRC e comparando-os com os valores transmitidos. Comprovada a correcção lógica da mensagem, é executada a função *USBProcessInMessage()* que efectua a tarefa respectiva.

4.2. Interface Gráfica

A implementação da IMD como meio interactivo para a operação da ECU exigiu o desenvolvimento de uma plataforma capaz de facilitar o *debug* e monitorização do sistema de controlo electrónico. A alteração dos parâmetros de afinação do motor, a leitura e monitorização em tempo real de certas variáveis, e a transferência de novas configurações de operação do motor são alguns exemplos da necessidade de criação de uma interface gráfica. Além disso, o potencial desenvolvimento de uma plataforma de testes e afinação do motor impõe indubitavelmente o acesso de forma facilitada a toda a informação gerada para posterior análise e processamento.

Fundamentadas as razões da implementação da interface gráfica, procede-se à explicação mais detalhada do seu processo de desenvolvimento. Na fase de criação do aspecto gráfica da interface recorreu-se à utilização do programa *Microsoft Visual Studio 2008* em conjunto com a consulta de bibliografia técnica adequada ([22][23][24]).

Na *Figura 24* pode ver-se o aspecto gráfico da janela principal no arranque da IMD, onde é possível observar directamente algumas das funcionalidades implementadas. O botão "*Connect"* permite efectuar a ligação com a ECU utilizando uma configuração pré-definida para a porta COM a ser aberta, ou definir outra porta e velocidade de comunicação disponíveis (o procedimento para a alteração dessa configuração será explicada adiante). A ocorrência de qualquer falha na abertura da porta COM é sinalizada através do surgimento de uma caixa de diálogo, reportando o erro associado. Em caso de sucesso, a Barra de Estado (parte inferior da janela) assinala a abertura da porta indicando o número correspondente. O cancelamento da ligação é efectuado pressionando o mesmo botão, que nessa altura apresenta o texto *"Disconnect"* a assinalar a função. Os restantes botões executam tarefas que requerem o estabelecimento prévio de uma ligação, uma vez que interagem com a ECU. Deste modo, a operação destes botões só fica disponível depois de estabelecida uma ligação com sucesso.

Figura 24 – Imagem da janela principal da IMD.

Relativamente aos botões contidos no grupo *"Test Modules"*, a sua funcionalidade é implícita: permitem testar a conectividade estabelecida entre os diferentes módulos. Cada módulo tem um botão distinto associado, permitindo a qualquer momento conhecer o estado de conexão desse módulo dentro do sistema. A informação obtida é exibida na Caixa de Texto situada por cima dos botões de teste, indicando o sucesso ou a falha da operação.

No grupo *"Obtain"*, cujas funções são igualmente de dedução imediata, são incluídas todas as operações referentes à obtenção dos valores actualizados de diversos parâmetros do motor. É possível observar o botão respeitante à obtenção do valor da pressão no depósito do combustível (em kilopascal), e a Caixa de Texto correspondente para exibição do valor obtido.

Resta ainda abordar a Barra de Menus que permite o encerramento da aplicação com a opção *"Exit"* no menu *"File"*, o acesso à janela de configuração da porta COM através da opção *"Port Settings"* no menu *"Settings"*, e a abertura da janela *"About"* presente no menu *"Help"*.

Na *Figura 25* encontra-se a imagem com o aspecto gráfico da janela de configuração da porta COM. Esta janela permite definir qual o número da porta a ser utilizada e o valor de *Baudrate* pretendido (a lista das portas disponíveis é gerada recorrendo a uma função própria do sistema).

Figura 25 – Imagem da janela de configuração da porta COM.

Por último, expõe-se na *Figura 26* a apresentação gráfica da janela *"About"* referente a esta aplicação. Contém informações gerais como a versão, o autor e uma breve descrição sobre o objectivo da aplicação.

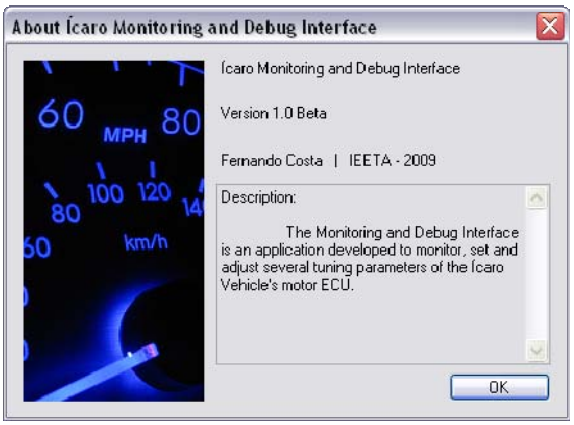


Figura 26 – Imagem da janela "*About*" da IMD.

Finaliza-se deste modo, a descrição da interface gráfica desenvolvida, do seu modo de operação e das potenciais funcionalidades a implementar. Simultaneamente, conclui-se o capítulo respeitante à exposição das características e funcionalidades da Interface de Monitorização e Diagnóstico, e ainda da solução desenvolvida para o protocolo de comunicação com a ECU.

Capítulo 5

5. Resultados e Discussão

Neste capítulo apresentam-se os resultados relativos à análise do funcionamento dos dois protocolos de comunicação implementados; o LIN e a interface USB. Essa demonstração é efectuada recorrendo a captações de imagens durante a operação da Interface de Monitorização e Diagnóstico. Através do envio de mensagens do tipo "*I'm Alive Request*" a partir do PC até cada módulo de controlo e a recepção da resposta (válida ou não), é possível verificar o estado de cada módulo e da rede de comunicação LIN. É efectuado ainda o pedido, ao respectivo módulo, do valor medido da pressão de combustível, sendo este valor enviado para o PC e apresentado na IMD. São igualmente testadas algumas situações de falha, como sejam a interrupção do meio de comunicação LIN entre dois módulos e a desconexão da interface USB.

Na *Figura 27* é possível visualizar o teste da comunicação entre o PC e o Módulo de Controlo Central, demonstrando o estabelecimento bem sucedido da comunicação através da interface USB. Na barra de estado observa-se a situação actual da conexão utilizando a porta virtual COM20.

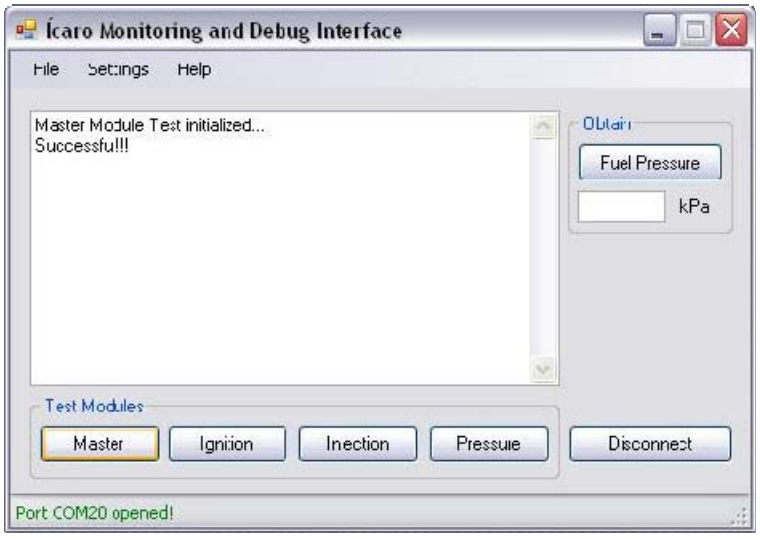


Figura 27 – Demonstração do correcto estabelecimento da comunicação através da interface USB.

Na *Figura 28* é possível observar o resultado do teste de comunicação com cada um dos restantes módulos, podendo constatar-se que todos se encontram correctamente ligados através do protocolo LIN. Foi pressionado o botão correspondente a cada módulo e consequentemente exibida a notificação do teste bem sucedido de todos eles.

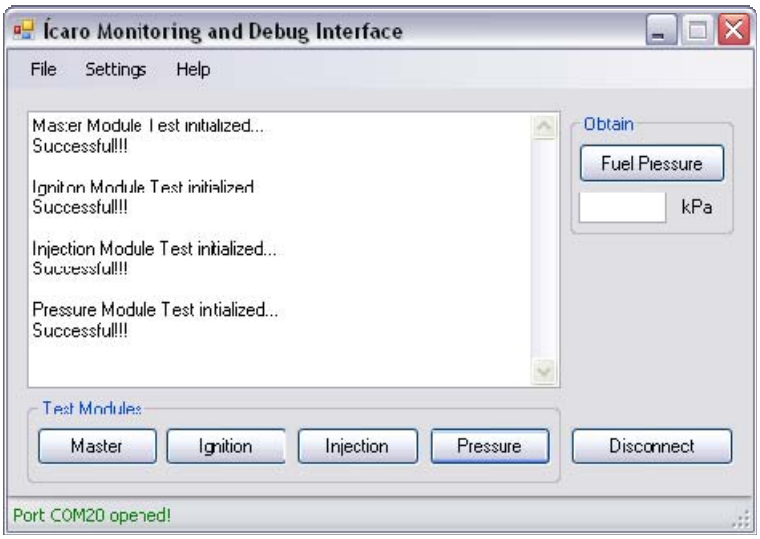


Figura 28 – Demonstração do funcionamento correcto do protocolo LIN e da conexão estabelecida entre todos os módulos.

Na figura seguinte exibe-se uma demonstração de aquisição do valor actual da pressão medido pelo sensor de pressão relativa do ar, existente no Módulo de Controlo da Pressão do Combustível. A medição da pressão do combustível foi efectuada utilizando o depósito pressurizado, ou seja, o resultado obtido representa a diferença entre a pressão medida e a pressão atmosférica (valor de referência). Para tal, foi utilizada uma bancada de ensaio para testar o funcionamento do Módulo de Controlo da Pressão do Combustível.

Na *Figura 29* encontra-se representada a aquisição do valor instantâneo da pressão no depósito de combustível. O valor apresentado de 297,22kPa encontra-se bastante próximo da pressão pretendida no interior do depósito (300kPa ou 3bar), comprovando o sucesso na implementação do controlo da mesma.

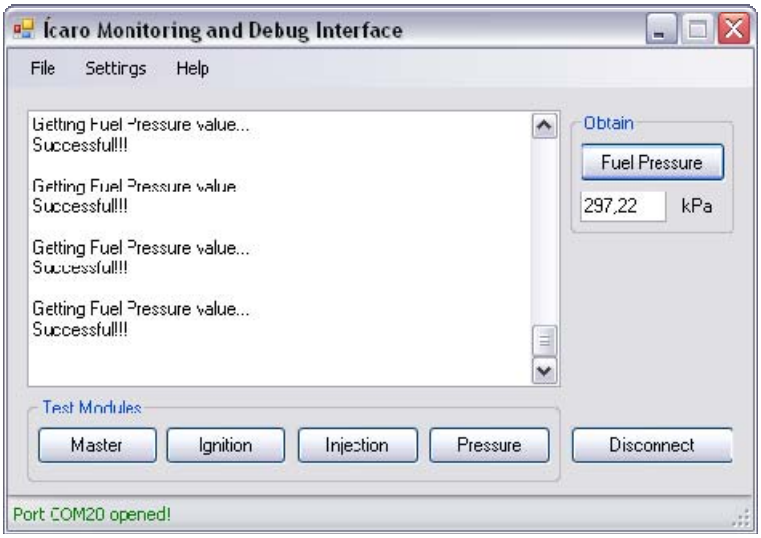


Figura 29 – Demonstração da aquisição do valor de pressão no depósito de combustível (valor adquirido próximo do pretendido).

Na *Figura 30* é possível observar os resultados da simulação de falha na rede de comunicação LIN. Neste teste, todos os módulos excepto o *Master*, foram desligados da rede LIN obtendo-se a notificação de falha nos testes respectivos.

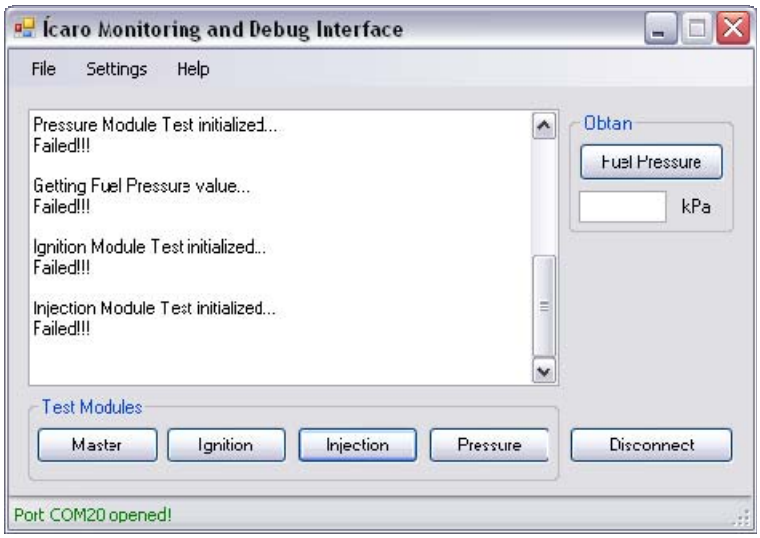


Figura 30 – Demonstração da simulação de falha no protocolo LIN.

Os resultados apresentados demonstram o funcionamento integral da rede de comunicação LIN implementando a conexão dos módulos de controlo. Igualmente se comprova o funcionamento da comunicação USB e a operacionalidade da Interface de Monitorização e Diagnóstico como potencial plataforma de teste, *debug* e configuração de todo o sistema.

De seguida são expostos os resultados quantitativos adquiridos durante a execução de testes de funcionamento dos protocolos implementados. Utilizando a IMD, realizaram-se dois testes distintos baseados no tipo de mensagens transmitidas: o primeiro através do envio de mensagens do tipo "*I'm Alive Request*" para cada módulo; o segundo através da requisição do valor da pressão do combustível ao Módulo de Controlo da Pressão do Combustível. Deste modo obtiveram-se valores concretos sobre a quantidade de mensagens perdidas no decorrer da execução dos dois protocolos.

A *Tabela 1* apresenta os resultados do primeiro teste, consistindo no envio de 100 mensagens (do tipo anteriormente mencionado) para cada módulo, indicando respectivamente a percentagem de mensagens falhadas em cada protocolo.

	Módulo de Controlo Central	Módulo de Controlo da Ignição	Módulo de Controlo da Injecção	Módulo de Controlo da Pressão
LIN	**-**	**7%**	**6%**	**6%**
USB	**2%**	**2%**	**3%**	**4%**

Tabela 1 – Resultados, em percentagem, da quantidade de mensagens falhadas em cada protocolo.

Na *Tabela 2* encontram-se os resultados experimentais do segundo teste realizado, consistindo em três ensaios para a aquisição do valor instantâneo da pressão do combustível. Em cada ensaio foram enviadas 100 mensagens registando-se a quantidade de falhas no processo de comunicação, e ainda a que protocolo era atribuído cada erro registado.

	Ensaio 1	Ensaio 2	Ensaio 3
LIN	**5%**	**7%**	**4%**
USB	**3%**	**3%**	**2%**

Tabela 2 – Resultados, em percentagem, da quantidade de erros registados em cada protocolo.

A partir dos resultados experimentais obtidos é possível inferir que existe concordância nos valores obtidos nos dois testes efectuados, reforçando a autenticidade dos mesmos. Relativamente ao primeiro teste, no protocolo LIN obteve-se uma percentagem média total de falhas igual 6,3%, e no protocolo USB obteve-se um valor homólogo igual a 2,8%. No segundo teste realizado obteve-se um valor médio total de erros dos três ensaios efectuados igual a 5,3% para o protocolo de comunicação LIN, e de 2,7% referente à comunicação USB. Considerando os resultados obtidos, no protocolo de comunicação observa-se uma percentagem média total de falhas na transmissão de mensagens superior ao protocolo USB implementado.

Relativamente à comunicação LIN, o tipo de mensagens enviadas nos testes apresentados são todas do tipo *Single Data Block*, variando o número de *Data Bytes* segundo a mensagem enviada (*e.g.* dois bytes na aquisição do valor da pressão e um byte no "*I'm Alive Request*"). O teste de envio de *Muti Block Data Messages* foi efectuado durante a fase de desenvolvimento do software de controlo da comunicação LIN, obtendo-se de forma análoga, sucesso na transmissão e recepção desse tipo de mensagens.

É essencial referir as dificuldades encontradas durante a implementação do protocolo LIN ao nível da SCI dos microcontroladores. Apesar das funcionalidades específicas para a implementação do LIN incluídas no microcontrolador utilizado no *Master* – que facilitaram de sobremaneira o trabalho a desenvolver – a correcta implementação de envio e recepção de mensagens revelou-se uma tarefa morosa.

Na interface de comunicação USB obtiveram-se igualmente resultados satisfatórios relativamente ao seu funcionamento. O protocolo de comunicação foi especificamente desenvolvido para o tipo de informação transmitida (resultando em algumas semelhanças com o formato das mensagens LIN), obrigando a uma transversalidade na implementação do mesmo. Os resultados apresentados comprovam o sucesso da operacionalidade da interface de comunicação USB, visto que todos os testes de transmissão e recepção de mensagens foram realizados executando com êxito a IMD a partir de um computador.

Capítulo 6

6. Conclusões e Desenvolvimentos Futuros

Neste capítulo procede-se à análise do trabalho desenvolvido face aos objectivos inicialmente propostos. Adicionalmente, são apresentadas propostas de desenvolvimentos futuros tendo por base o trabalho efectuado e exposto ao longo deste documento. Consiste, portanto, num capítulo essencial para o esclarecimento do sucesso e das falhas na execução dos propósitos indicados para o sistema final pretendido, bem como do esforço e dedicação atribuídos a esta dissertação.

Os objectivos inicialmente propostos, tal como apresentados em *1.2*, incluíam a execução do trabalho a realizar em duas fases. Na fase inicial exigia-se o levantamento do estado da arte das áreas afectas ao projecto, a análise do problema considerando o binómio custo/benefício da implementação de uma solução distribuída versus centralizada. Na finalização desta fase elaborou-se um documento sobre o estudo e pesquisa efectuados, concluindo-se com uma apresentação sobre os mesmos. A importância desta fase inicial revelou-se fundamental na aquisição de conhecimentos relacionados com o projecto, nomeadamente nos motores de combustão interna e sistemas distribuídos. Potenciou a familiarização de conceitos e o delineamento e estruturação do trabalho a desenvolver.

A segunda fase do projecto consistia na etapa *hands-on* estando dividida em quatro pontos. O ponto inicial envolvia a análise e estudo dos módulos electrónicos de controlo já desenvolvidos, bem como a finalização e validação técnica dos mesmos. É essencial referir que a execução desta tarefa prolongou-se substancialmente na calendarização de todo o projecto. A falta de alguns componentes, a detecção de falhas em pistas da PCB e o teste de todo hardware envolveram a dedicação de tempo e esforços acrescidos. Verificaram-se algumas avarias de componentes cuja detecção originou implicações significativas na calendarização do projecto (*e.g.* substituição do microcontrolador PIC18F4585 e de um LIN *Transceiver* MCP201). Foi necessário da mesma forma finalizar a montagem de outros componentes e fichas de conexão, bem como efectuar a ligação física entre os módulos permitindo a implementação e teste da rede de comunicação LIN. O estudo dos esquemas eléctricos e o envolvimento necessários para a compreensão do funcionamento dos módulos contribuíram igualmente para essa dedicação.

Devido ao facto do desenvolvimento do software de controlo electrónico pressupor o conhecimento e domínio integral do modo de funcionamento da solução centralizada, a sua implementação revelou-se uma tarefa extensa e árdua. É imprescindível considerar que a solução anterior é o resultado de vários anos de trabalho e *know-how* acumulado desde a existência do projecto Ícaro. Deste modo, completados os teste de hardware e validação dos módulos electrónicos, procedeu-se à implementação da rede de comunicação LIN, delegando para uma fase posterior a dedicação ao desenvolvimento do software de controlo de cada módulo.

Através dos resultados apresentados é possível comprovar a implementação funcional do protocolo de comunicação LIN com sucesso. Efectuaram-se ensaios de transmissão e recepção dos vários tipos de mensagens utilizadas, realizando alterações ao formato das mesmas com vista a adequar o protocolo ao tipo de informação utilizada no sistema. Os resultados experimentais obtidos permitem concluir que a percentagem da perda de mensagens no protocolo de comunicação LIN é inferior a 6%. A justificação deste valor poderá encontrar-se no facto de serem detectados erros durante o processamento das mensagens pela máquina de estados implementada. Desenvolveu-se, desta forma, a base operacional de suporte às comunicações LIN entre os módulos de controlo electrónico.

A implementação da interface de comunicação USB instalada no Módulo de Controlo Central permitiu a execução de testes ao protocolo de comunicação LIN e consequentemente à operacionalidade dos restantes módulos. Essa implementação exigiu o desenvolvimento de uma Interface de Monitorização e Diagnóstico com o propósito de facilitar e auxiliar no supervisionamento e *debug* do sistema. Essa aplicação foi desenvolvida em VB.NET, representando uma dificuldade e um desafio acrescidos (associados à aprendizagem dessa nova linguagem), e novamente introduzindo atrasos na execução do projecto. A falta de conhecimentos adquiridos e familiarização na programação em VB.NET implicou que o desenvolvimento do código produzido não estivesse optimizado a essa linguagem. Desta irregularidade surgiram contratempos e falhas esporádicas existentes na IMD, nomeadamente a ocorrência de situações de bloqueio e interrupção de comunicação com a centralina. No entanto, tentaram-se resolver atempadamente os principais erros de forma a não comprometer gravemente o decurso do projecto. Assim, recorrendo aos resultados expostos anteriormente verifica-se que a percentagem média de erros na comunicação USB é inferior a 3%, comprovando-se o sucesso operacional do protocolo e da Interface de Monitorização e Diagnóstico.

O desenvolvimento da IMD e da comunicação USB inclui-se no terceiro objectivo da segunda fase do trabalho, que consistia na execução e teste do sistema distribuído de controlo electrónico do motor. Tal como referido anteriormente, optou-se inicialmente pelo desenvolvimento do protocolo de comunicação LIN, delegando para uma fase posterior a implementação do software de controlo dos módulos. Deste modo, apenas o Módulo de Controlo da Pressão do Combustível possui essa componente desenvolvida, além da implementação não testada do canal de sincronização dedicado entre o Módulo de Controlo da Ignição e o Módulo de Controlo da Injecção. Nos restantes módulos, apesar do empenho

dedicado a terminar essa tarefa, não foi possível desenvolver totalmente o software de controlo electrónico. Devido à extensão e complexidade do software da solução centralizada que serviu de suporte à realização dessa tarefa, o propósito de produzir uma solução distribuída totalmente testada e funcional não foi alcançado.

É importante salientar, no entanto, que apesar do objectivo maior não ter sido alcançado, foi criada toda a estrutura de comunicação que servirá de base à solução distribuída de controlo de motores de combustão interna. É essencial analisar e referir o benefício da implementação do controlo electrónico de um motor de combustão interna sob a forma de um sistema distribuído. Além de todo o conjunto de vantagens referidas em *1.4.2* sobre os sistemas desenvolvidos como uma solução distribuída, note-se que neste caso as potencialidades são imediatas: todo o sistema desenvolvido para o motor mono cilíndrico pode ser extrapolado para um motor multi-cilíndrico, procedendo-se à sua reprodução e adaptação correspondente ao número de cilindros do motor a utilizar. Evita-se desta forma, a concentração do sistema de controlo electrónico numa única unidade, facultando alterações e intervenções de forma mais fácil, rápida e com menos custos (implementação modular).

As propostas de desenvolvimentos futuros cingem-se à continuidade do trabalho realizado no sentido de produzir a solução distribuída proposta inicialmente. Para tal, será necessário produzir o software de controlo em falta nos respectivos módulos e elaborar uma configuração de ensaio para teste e afinação do motor. Esta tarefa exigirá complementar a IDM com os restantes parâmetros de ajuste, controlo e leitura do estado do motor. Seguindo a tendência actual da crescente utilização de comunicações sem fios, poder-se-á eventualmente substituir a interface USB por um módulo de comunicação *wireless*, nomeadamente *Bluetooth* ou *ZigBee*. Esta alteração permitirá efectuar o *debug* e monitorização do motor sem a necessidade de recorrer a cabos, e sobretudo eliminar a necessidade de acesso físico à ECU (quando instalado no motor e no veículo).

A quarta e última etapa do projecto – materializada neste documento escrito – consistia na redacção da dissertação sobre o trabalho desenvolvido. Assim, ao longo deste documento apresentaram-se os estudos, análises, implementações, desenvolvimentos, resultados e conclusões sobre todo o trabalho proposto. Espera-se portanto, que os seus leitores obtenham um esclarecimento o mais detalhado possível relativamente ao tempo, dedicação e expectativas criadas ao longo da execução deste projecto.

Referências Bibliográficas

1. ALMEIDA, L. **Disciplina de Sistemas de Tempo Real**. DETI/UA. Aveiro. 2007.

2. MULLENDER, S. **Distributed Systems / edited by Sape Mullender**. 2nd. ed. New York: ACM Press, 1994.

3. LAMPORT, L. Distribution Email. **Leslie's Lamport Home Page**, 28 Maio 1987. Disponivel em: <http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html>. Acesso em: 16 Fevereiro 2009.

4. WEITZMAN, C. **Distributed Micro/Minicomputer Systems:** Structure, Implementation, and Application. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1980.

5. KOPETZ, H. **Real-Time Systems:** Design Principles for Distributed Embedded Applications. Boston: Kluwer Academic Publisher, 1997.

6. HEINECKE, HARALD - BMW GROUP. **Automotive System Design - Challenges and Potential**. Design, Automation and Test in Europe Conference and Exhibition. Munich: IEEE. 2005. p. 2.

7. TURLEY, J. The Two Percent Solution. **Embedded Systems Design – United Business Media**, 18 Dezembro 2002. Disponivel em: <http://www.embedded.com/shared/printableArticle.jhtml?articleID=9900861>. Acesso em: 16 Fevereiro 2009.

8. MAYER, E. **Serial Bus Systems in the Automobile Part 3: Simple and cost-effective data exchange in the automobile with LIN**. Vector Software. Stuttgart. 2006.

9. JURGEN, R. K. **Automotive Electronics Handbook**. 2nd. ed. New york: McGraw-Hill, 1999.

10. VECTOR. **Vector**. Disponivel em: <www.vector.com>. Acesso em: Fevereiro 2009.

11. NAVET, N. et al. Trends in Automotive Communication Systems. **Proceedings Of The IEEE**, Vandoeuvre-lés-Nancy, France, 93, June 2005. 20.

12. NOLTE, T.; HANSSON, H.; LO BELLO, L. **Automotive Communications - Past, Current and Future**. 10th IEEE Conference on Emerging Technologies and Factory Automation. Catania, Italy: IEEE. 2005. p. 8.

13. GABRIEL, C. **Integrating Sensor Devices in a LIN bus network**. 26th International Spring Seminar on Electronics Technology. Stari Lesni, Slovak Republic: [s.n.]. 2003. p. 4.

14. RUFF, M. **Evolution of Local Interconnect Network (LIN)**. Motorola, Inc. Austin, Texas, p. 8. 2003.

15. LIN CONSORTIUM. **LIN – Local Interconnect Network**. Disponivel em: <http://www.lin-subbus.org>. Acesso em: 2009.

16. RUFF, M. **Evolution of Local Interconnect Network (LIN) Solutions**. Vehicular Technology Conference. Orlando, Florida: [s.n.]. Outubro 2003. p. 3382 - 3389.

17. BOSCH. **Bosch Automotive Handbook**. 7th. ed. Chichester: John Wiley & Sons, 2007.

18. RIBBENS, W. B. **Understanding Automotive Electronics**. 6th. ed. New York: Newnes, 2003.

19. PULKRABEK, W. W. **Engineering Fundamentals Of The Internal Combustion Engine**. New Jersey: Prentice Hall, 1997.

20. MICROCHIP TECHNOLOGY INC. **PICs 18F4585/18F485/18F6722/12F683 Datasheets**. Microchip. Shanghai.

21. FTDI LTD. **UM245R USB-Parallel FIFO Development Module Datasheet**. FTDI. Glasgow, UK, p. 19. 2005. (Version 1.02).

22. HALVORSON, M. **Microsoft Visual Basic.NET Passo a Passo**. Portugal: McGrawHill, 2002.

23. MICROSOFT. Visual Basic Developer Center. **MSDN**. Disponivel em: <http://msdn.microsoft.com/en-us/vbasic/ee658249.aspx>. Acesso em: Julho 2009.

24. STARTVBDOTNET.COM. VB Language. **Startvbdotnet**. Disponivel em: <http://www.startvbdotnet.com/language/default.aspx>. Acesso em: Julho 2009.

25. MAYER, E. **Serial Bus Systems in the Automobile Part 1: Motivation, advantages, tasks and architecture of serial bus systems in the automobile**. Vector Software. Stuttgart. 2006.